

UUU	UUU	EEEEEEEEEEEEEEEE	TTTTTTTTTTTTTTTT	PPPPPPPPPPPPPP	
UUU	UUU	EEEEEEEEEEEEEEEE	TTTTTTTTTTTTTTTT	PPPPPPPPPPPPPP	
UUU	UUU	EEEEEEEEEEEEEEEE	TTTTTTTTTTTTTTTT	PPPPPPPPPPPPPP	
UUU	UUU	EEE	TTT	PPP	PPP
UUU	UUU	EEE	TTT	PPP	PPP
UUU	UUU	EEE	TTT	PPP	PPP
UUU	UUU	EEE	TTT	PPP	PPP
UUU	UUU	EEE	TTT	PPP	PPP
UUU	UUU	EEE	TTT	PPP	PPP
UUU	UUU	EEEEEEEEEEEEEEEE	TTT	PPPPPPPPPPPPPP	
UUU	UUU	EEEEEEEEEEEEEEEE	TTT	PPPPPPPPPPPPPP	
UUU	UUU	EEEEEEEEEEEEEEEE	TTT	PPPPPPPPPPPPPP	
UUU	UUU	EEE	TTT	PPP	
UUU	UUU	EEE	TTT	PPP	
UUU	UUU	EEE	TTT	PPP	
UUU	UUU	EEE	TTT	PPP	
UUU	UUU	EEE	TTT	PPP	
UUU	UUU	EEE	TTT	PPP	
UUU	UUU	EEE	TTT	PPP	
UUUUUUUUUUUUUUUU	UUUUUUUUUUUUUUUU	EEEEEEEEEEEEEEEE	TTT	PPP	
UUUUUUUUUUUUUUUU	UUUUUUUUUUUUUUUU	EEEEEEEEEEEEEEEE	TTT	PPP	
UUUUUUUUUUUUUUUU	UUUUUUUUUUUUUUUU	EEEEEEEEEEEEEEEE	TTT	PPP	

UU	UU	EEEEEEEEEE	TTTTTTTTTT	PPPPPPPP	HH	HH	AAAAAA	SSSSSSSS	000000	000000
UU	UU	EEEEEEEEEE	TTTTTTTTTT	PPPPPPPP	HH	HH	AAAAAA	SSSSSSSS	000000	000000
UU	UU	EE	TT	PP	PP	HH	AA	SS	00	00
UU	UU	EE	TT	PP	PP	HH	AA	SS	00	00
UU	UU	EE	TT	PP	PP	HH	AA	SS	00	0000
UU	UU	EE	TT	PP	PP	HH	AA	SS	00	0000
UU	UU	EEEEEEEEEE	TT	PPPPPPPP	HHHHHHHHHH	AA	AA	SSSSSS	00	00
UU	UU	EEEEEEEEEE	TT	PPPPPPPP	HHHHHHHHHH	AA	AA	SSSSSS	00	00
UU	UU	EE	TT	PP	HH	HH	AAAAAAAAAA	SS	0000	00
UU	UU	EE	TT	PP	HH	HH	AAAAAAAAAA	SS	0000	00
UU	UU	EE	TT	PP	HH	HH	AA	SS	00	00
UU	UU	EE	TT	PP	HH	HH	AA	SS	00	00
UUUUUUUUUU	EEEEEEEEEE	TT	PP	HH	HH	HH	AA	SSSSSSSS	000000	000000
UUUUUUUUUU	EEEEEEEEEE	TT	PP	HH	HH	HH	AA	SSSSSSSS	000000	000000

LL	IIIIII	SSSSSSSS
LL	IIIIII	SSSSSSSS
LL	II	SS
LL	II	SS
LL	II	SS
LL	II	SS
LL	II	SSSSSS
LL	II	SSSSSS
LL	II	SS
LL	II	SS
LL	II	SS
LL	II	SS
LLLLLLLLLL	IIIIII	SSSSSSSS
LLLLLLLLLL	IIIIII	SSSSSSSS

(2)	107	Declarations
(3)	206	Read-Only Data
(9)	400	Process Info Data Structure
(10)	501	Read/Write Data
(23)	925	RMS-32 Data Structures
(24)	986	Local area for ID descriptor
(25)	1008	Local area for Process Termination Subroutine
(26)	1057	Local area for create command procedure w/ parameters
(27)	1082	Local area for Create Process Subroutine
(28)	1116	Local area for TPARSE and Get/Parse Routine
(29)	1150	Main Program
(30)	1670	Abort TSTCNTRL Subroutine
(31)	1760	Timer Expiration Subroutine
(32)	1882	Process Termination Subroutine
(34)	2441	End-of-file Subroutine
(35)	2544	Filename found Subroutine
(36)	2673	Create Process Subroutine
(37)	3166	Create image procedure
(38)	3223	Create image procedure with parameters
(39)	3319	Create command procedure
(40)	3381	Create command procedure with parameters
(41)	3585	Create Process-name
(42)	3697	Make temporary file name
(43)	3807	Assignment Expression Subroutine
(44)	3948	Assignment Routine for Process Name
(45)	4011	Assignment Routine for Starting Sentinel
(46)	4071	Assignment Routine for Ending Sentinel
(47)	4130	Assignment Routine Changing Comment Flag
(48)	4206	Assignment Routine for Log File Name
(49)	4294	Assignment Routine for Setting Watchdog Timer
(50)	4377	Assignment Routine for Changing the Parallel Count
(51)	4485	Get and Parse Record
(52)	4779	State Table for Data File
(64)	5281	Initialize data structure
(65)	5367	Expand region for more data areas
(66)	5479	Abort Detached Processes
(67)	5588	Nuke Detached Processes AST
(68)	5689	Process Termination AST
(69)	5853	Find Process Slot
(70)	5955	Maximum Timer Expiration AST
(71)	6015	System Service Exception Handler
(73)	6248	RMS Error Handler
(74)	6376	CTRL/C Handler
(75)	6461	Exit Handler

```
0000 1 .TITLE TSTCNTRL TEST PACKAGE CONTROL PROGRAM
0000 2 .IDENT 'V04-000'
0000 3 .ENABLE SUPPRESSION
0000 4
0000 5 *****
0000 6 *
0000 7 * COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
0000 8 * DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
0000 9 * ALL RIGHTS RESERVED.
0000 10 *
0000 11 * THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
0000 12 * ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
0000 13 * INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
0000 14 * COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
0000 15 * OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
0000 16 * TRANSFERRED.
0000 17 *
0000 18 * THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0000 19 * AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
0000 20 * CORPORATION.
0000 21 *
0000 22 * DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
0000 23 * SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
0000 24 *
0000 25 *
0000 26 *****
0000 27
0000 28
0000 29 ++
0000 30 FACILITY:
0000 31 UETP -- TEST PACKAGE CONTROL PROGRAM
0000 32
0000 33 ABSTRACT:
0000 34 This program will serve as the controlling program between various
0000 35 test programs. It will create the test programs (either .EXE or
0000 36 .COM) as detached processes and run them either synchronously or
0000 37 asynchronously.
0000 38
0000 39 ENVIRONMENT:
0000 40 This program will run in user access mode, with interrupts enabled
0000 41 at all times. This program requires the following privileges and
0000 42 quotas:
0000 43 either SETPRIV or (DETACH and WORLD)
0000 44
0000 45
0000 46 AUTHOR: Brian A. Axtell, CREATION DATE: December, 1982
0000 47
0000 48 MODIFIED BY:
0000 49
0000 50 V03-011 RNH0003 Richard N. Holstein, 07-Apr-1984
0000 51 Adapt to security change which put mailbox logical names in
0000 52 job tables rather than group tables. When reporting failure
0000 53 of some process, always return full text of error code. Fix
0000 54 bug which reported wrong status when process creation failed.
0000 55 Indent all of copied log file lines uniformly.
0000 56
0000 57 V03-010 RNH0002 Richard N. Holstein, 22-Mar-1984
```

```
0000 58 : Fix bug which thought that turning off a test was a syntax
0000 59 : error.
0000 60 :
0000 61 : V03-009 BAA0007 Brian A. Axtell, 17-Jan-1984
0000 62 : Added capability to change the parallel count number
0000 63 : dynamically from within a data file.
0000 64 : Fixed other assignments so that they don't write messages
0000 65 : if wrapping around.
0000 66 : Added "EXIT" keyword, so one could end with it instead of EOF.
0000 67 : Added the capability of using abbreviations of keywords.
0000 68 : Fixed bug with ABORT_PROCESSES. The FORCEX was stopping images
0000 69 : so fast that the Process Termination AST was being called and
0000 70 : destroying the queue before all the processes could be stopped.
0000 71 : The fix was to calculate the offset to the next process before
0000 72 : aborting the current process.
0000 73 :
0000 74 : V03-008 BAA0006 Brian A. Axtell, 03-Jan-1984
0000 75 : Fixed problem with getting error messages when already
0000 76 : in ABORT sequence.
0000 77 :
0000 78 : V03-007 RNH0001 Richard N. Holstein, 19-Dec-1983
0000 79 : Use LIB$SIGNAL or $PUTMSG throughout, instead of LIB$PUT_OUTPUT.
0000 80 :
0000 81 : V03-006 BAA0005 Brian A. Axtell, 16-Aug-1983
0000 82 : Set flag so beginning and ending sentinels are always
0000 83 : printed out.
0000 84 :
0000 85 : V03-005 BAA0004 Brian A. Axtell, 15-Jul-1983
0000 86 : Added <CR><LF> before prompt in interactive mode.
0000 87 : Changed locally defined symbol to global symbol
0000 88 : (PRCSM_LOGIN) for $CREPRC
0000 89 :
0000 90 : V03-004 BAA0003 Brian A. Axtell, 21-Mar-1983
0000 91 : Changed attributes of the log file FAB's. Put a header
0000 92 : of blanks when copying records from log file. Change
0000 93 : labels of the inputs from the CLD file.
0000 94 :
0000 95 : V03-003 BAA0002 Brian A. Axtell, 17-Jan-1983
0000 96 : Changed process termination routine so that it reports
0000 97 : errors when a process aborts abnormally.
0000 98 :
0000 99 : V03-002 BAA0001 Brian A. Axtell, 16-Dec-1982
0000 100 : Changed the way PRCINFO$_FLAGS was defined.
0000 101 :
0000 102 : V03-001 LDJ0001 Larry D. Jones, 16-Dec-1982
0000 103 : Added a $ to the TCNTRLDEF macro call.
0000 104 :
0000 105 : ++
```

49

53

40

54

45

```
0000 107      .SBTTL Declarations
0000 108
0000 109 ; INCLUDE FILES:
0000 110
0000 111 ; MACROS:
0000 112
0000 113      $$$DEF      ; for system services
0000 114      $DEVDEF     ; device charac. via RMS
0000 115      $DVIDEF    ; for $GETDVI values
0000 116      $DCDEF     ; for device characteristics
0000 117      $JPIDEF    ; for GETJPI system service
0000 118      $LNMDDEF   ; Logical name services codes
0000 119      $PQLDEF    ; for CREPRC system service
0000 120      $CHFDEF    ; Condition handler frame definition
0000 121      $DSCDEF    ; include offsets for descriptor
0000 122      $DIBDEF    ; Device Information Block
0000 123      $SHRDEF    ; Shared messages
0000 124      $STSDEF    ; Status return
0000 125      $ACCDDEF   ; accounting messages
0000 126      $PRVDEF    ; privilege offset values
0000 127      $NAMDEF    ; to get NAM offsets for parse
0000 128      $LIBDEF    ; to get LIB error symbols
0000 129      $TPADEF    ; to get TPARSE symbols
0000 130      $PRCDEF    ; get masks for CREPRC
0000 131      $UETPDEF   ; UETP symbols
0000 132
0000 133      ; now define the various offset used by the TSTCNTRL. These include
0000 134      ; TCNTRL$, PRCINFO$, PARSE$, CASE$, CRTPRC$, and ASSIGN$.
0000 135
0000 136      $TCNTRLDEF     ; definitions for the TSTCNTRL
0000 137
0000 138 ; EQUATED SYMBOLS:
0000 139
0000 140      ; Facility number definitions:
0000 141
0000 142      TCNTRL_K = 195      ; define the facility code
0000 143      RMS_K = 1
0000 144
0000 145      ; SHR message definitions:
0000 146
0000 147      TCNTRL = TCNTRL_K$STSSV_FAC_NO      ; Define the TCNTRL facility code
0000 148
0000 149      TCNTRL$_ABEND    = TCNTRL!SHR$_ABEND    ; Define the TCNTRL message codes
0000 150      TCNTRL$_ABORT    = TCNTRL!SS$_ABORT
0000 151      TCNTRL$_CONTROL  = TCNTRL!SS$_CONTROL
0000 152      TCNTRL$_BEGIN    = TCNTRL!SHR$_BEGIN
0000 153      TCNTRL$_ENDED    = TCNTRL!SHR$_ENDED
0000 154      TCNTRL$_OPENIN  = TCNTRL!SHR$_OPENIN
0000 155      TCNTRL$_TEXT    = TCNTRL!SHR$_TEXT
0000 156
0000 157      ; set up definitions
0000 158
0000 159      ; define the maximum number of processes which can run concurrently
0000 160
0000 161      NUMB_OF_SLOTS = 50      ; number of slots in one structure
0000 162      PAGE_SIZE = 512        ; number of bytes in a page
0000 163
```

```
0000 164
0000 165      ; misc. assignments
0000 166
00000084 0000 167      RCRD_SIZE = 132      ; size of text strings
00000100 0000 168      LARGE_BUFFER = 256      ; size of large output strings
00000009 0000 169      NAME_SIZE = 9      ; max size of a filename
00000054 0000 170      MAILBOX_SIZE = 84      ; size of AST mailbox
0000 171
00000004 0000 172      NBR_HDR_BLNKS = 4      ; number of blanks in header of
0000 173      ; output records
0000 174
0000 175      ; declarations of filespec information
0000 176
00000009 0000 177      FILENAME_LENGTH = 9      ; max length of VMS filename
00000004 0000 178      FILE_EXT_LENGTH = 4      ; size of VMS file extension (inc .)
0000000F 0000 179      PROCESS_NAME_SIZE = 15      ; max length of VMS processname
0000 180
0000 181      ; declarations of information for TSTCNTRL ID number and process name
0000 182
00000004 0000 183      ID_STR_SIZE = 4      ; number of digits in ID string
000000C2 0000 184      ID_OVERHEAD_SIZE = 2      ; underscore and blank
00000009 0000 185      PNAME_FILNAM_SIZE = PROCESS_NAME_SIZE -
0000 186      - ID_STR_SIZE - ID_OVERHEAD_SIZE      ; largest size the filename can
0000 187      ; be to be put into the process-
0000 188      ; name. Anything larger is truncat
0000 189
00000005 0000 190      MAX_TMP_LOG_NAM = FILENAME_LENGTH -
0000 191      - ID_STR_SIZE      ; maximum size of extracted
0000 192      ; filename for temp. log files
0000 193
0000 194      ; declarations of timer ID's and delta time for waiting for aborted
0000 195      ; process to terminate
0000 196
00000001 0000 196      MAXTIME_ID = 1      ; ID for MAXTIME
00000002 0000 197      ABRT_TIMER_ID = 2      ; ID for the abort timer
0000 198
0000000F 0000 199      ABRT_DELT_SCNDS = 15      ; set timer for 15 seconds
0000 200
0000 201      ; define symbols for carriage-return / line-feed
0000 202
0000000D 0000 203      CR = ^XOD      ; ascii code for carriage rtn
0000000A 0000 204      LF = ^XOA      ; ascii code for line-feed
```

TSTCNTRL
V04-000

TEST PACKAGE CONTROL PROGRAM
Read-On'y Data

H 1

16-SEP-1984 01:30:05 VAX/VMS Macro V04-00
5-SEP-1984 04:26:03 [UETP.SRC]UETPHAS00.MAR;1

Page 5
(3)

```

      0000 206      .SBTTL Read-Only Data
00000000 207      .PSECT RODATA,NOEXE,NOWRT,PAGE
      0000 208
      0000 209      ; prompt for interactive TSTCNTRL
      0000 210
      0000 211 TCNTRL_PRMT:      ; prompt for interactive
OD 0000 212      .BYTE CR
OA 0001 213      .BYTE LF
20 3E 4C 52 54 4E 43 54 0002 214      .ASCII /TCNTRL> /
      000A 215
      000A 216 TCNTRL_PRMT_SIZ = . - TCNTRL_PRMT      ; size of prompt
      000A 217
      000A 218
      000A 219      ; quadword delta time for the abort process timer
      000A 220
      000A 221 PROC_TERM_DELT:
FFFFFFFF F70F2E80 000A 222      .LONG -10*1000*1000*ABRT_DELT_SCNDS,-1 ; wait time for aborted proc.
      0012 223
      0012 224
      0012 225      ; data structures so that mailboxes which serve as SYS$INPUT to the
      0012 226      ; processes we create are created in our group logical name table.
      0012 227
      0012 228 LNMPRCDIR:      ; Table name to force mbx logicals...
52 50 24 4D 4E 4C 0000001A'010E0100' 0012 229      .ASCID /LNMS$PROCESS_DIRECTORY/ ; ...to appear in a group table
54 43 45 52 49 44 5F 53 53 45 43 4F 0020
      59 52 4F 002C
      002F 230
      002F 231 LNMTMPMBX:      ; Logical name which tells $CREMBX...
45 54 24 4D 4E 4C 00000037'010E0000' 002F 232      .ASCID /LNMS$TEMPORARY_MAILBOX/ ; ...where to put mbx logical names
4C 49 41 4D 5F 59 52 41 52 4F 50 4D 003D
      58 4F 42 0049
      004C 233
      004C 234 LNMITMLST:      ; $CREMBX ITMLST naming where mbx...
      004C 235      .WORD LNMGRPLEN,LNMS_STRING ; ...logical name is to be defined
00000000'000004FE' 0050 236      .ADDRESS LNMGRPNUM,0
      00000000 0058 237      .LONG 0
```

```
005C 239 ; space for $GETJPI to get the quotas of the parent process
005C 240
005C 241 QUOTA_LIS:
0004 005C 242 .WORD 4
0409 005E 243 .WORD JPIS ASTLM
0000045F' 0060 244 .ADDRESS ASTLM
00000000 0064 245 .LONG 0
0004 0068 246 .WORD 4
0310 006A 247 .WORD JPIS BIOLM
00000464' 006C 248 .ADDRESS BIOLM
00000000 0070 249 .LONG 0
0004 0074 250 .WORD 4
031A 0076 251 .WORD JPIS BYTLM
00000469' 0078 252 .ADDRESS BYTLM
00000000 007C 253 .LONG 0
0004 0080 254 .WORD 4
040D 0082 255 .WORD JPIS CPULIM
0000046E' 0084 256 .ADDRESS CPULM
00000000 0088 257 .LONG 0
0004 008C 258 .WORD 4
0313 008E 259 .WORD JPIS DIOLM
00000473' 0090 260 .ADDRESS DIOLM
00000000 0094 261 .LONG 0
0004 0098 262 .WORD 4
0320 009A 263 .WORD JPIS ENQLM
00000478' 009C 264 .ADDRESS ENQLM
00000000 00A0 265 .LONG 0
0004 00A4 266 .WORD 4
040F 00A6 267 .WORD JPIS FILLM
0000047D' 00A8 268 .ADDRESS FILLM
00000000 00AC 269 .LONG 0
0004 00B0 270 .WORD 4
040E 00B2 271 .WORD JPIS PGFLQUOTA
00000482' 00B4 272 .ADDRESS PGFLQUOTA
00000000 00B8 273 .LONG 0
0004 00BC 274 .WORD 4
0408 00BE 275 .WORD JPIS PRCLM
00000487' 00C0 276 .ADDRESS PRCLM
00000000 00C4 277 .LONG 0
0004 00C8 278 .WORD 4
0410 00CA 279 .WORD JPIS TQLM
0000048C' 00CC 280 .ADDRESS TQLM
00000000 00D0 281 .LONG 0
0004 00D4 282 .WORD 4
0402 00D6 283 .WORD JPIS WSQUOTA
00000496' 00D8 284 .ADDRESS WSQUOTA
00000000 00DC 285 .LONG 0
0004 00E0 286 .WORD 4
0309 00E2 287 .WORD JPIS PRIB
0000044E' 00E4 288 .ADDRESS BASPRI
00000000 00E8 289 .LONG 0
00000000 00EC 290 .LONG 0
```

```
00F0 292 ; error messages
00F0 293
00F0 294 ABORT_MSG:
00F0 295 .ASCII /TSTCNTRL beginning abort sequence. All active process to be stopped
54 4E 43 54 53 54 000000F8'010E0000' 00FE
67 6E 69 6E 6E 69 67 65 62 20 4C 52 010A
65 75 71 65 73 20 74 72 6F 62 61 20 0116
74 63 61 20 6C 6C 41 20 2E 65 63 6E 0122
20 73 73 65 63 6F 72 70 20 65 76 69 012E
65 70 70 6F 74 73 20 65 62 20 6F 74 013A
2E 64 013C
013C 296
013C 297 TCNTRL_ABORT_MSG:
013C 298 .ASCII /This process was aborted by the TSTCNTRL./
70 20 73 69 68 54 00000144'010E0000' 014A
61 20 73 61 77 20 73 73 65 63 6F 72 0156
68 74 20 79 62 20 64 65 74 72 6F 62 0162
2E 4C 52 54 4E 43 54 53 54 20 65 016D
016D 299
016D 300 CNTRLMSG:
016D 301 .ASCII \Aborted via a user CTRL/C.\
72 65 73 75 20 61 20 61 69 76 20 64 017B
2E 43 2F 4C 52 54 43 20 0187
018F 302
018F 303 FILNAM_ERR:
018F 304 .ASCII /FILNAM data file is invalid or was not found./
4D 41 4E 4C 49 46 00000197'010E0000' 019D
69 20 65 6C 69 66 20 61 74 61 64 20 01A9
72 6F 20 64 69 6C 61 76 6E 69 20 73 01B5
75 6F 66 20 74 6F 6E 20 73 61 77 20 01C1
2E 64 6E 01C4
01C4 305
01C4 306 REPORT_ERR:
01C4 307 .ASCII /Invalid REPORT, default used./
69 6C 61 76 6E 49 000001CC'010E0000' 01D2
65 64 20 2C 54 52 4F 50 45 52 20 64 01DE
2E 64 65 73 75 20 74 6C 75 61 66 01E9
01E9 308
01E9 309 DATSTRUC_ERR:
01E9 310 .ASCII /Internal data structure corrupted, returned process not found./
6E 72 65 74 6E 49 000001F1'010E0000' 01F7
75 72 74 73 20 61 74 61 64 20 6C 61 0203
70 75 72 72 6F 63 20 65 72 75 74 63 020F
65 6E 72 75 74 65 72 20 2C 64 65 74 021B
6F 6E 20 73 73 65 63 6F 72 70 20 64 0227
2E 64 6E 75 6F 66 20 74 022F
022F 311
022F 312 BAD_STATUS:
022F 313 .ASCII/The process -!AS- returned a final status of:/
72 70 20 65 68 54 00000237'010E0000' 023D
20 20 53 41 21 20 20 73 73 65 63 6F 0249
66 20 61 20 64 65 6E 72 75 74 65 72 0255
20 73 75 74 61 74 73 20 6C 61 6E 69 0261
3A 66 6F 0264
0264 314
0264 315 NO_CRT_PRC:
0272 316 .ASCII \Could not create the -!AS- process, status returned was:\
6F 72 70 20 20 53 41 21 20 20 65 68 027E
73 75 74 61 74 73 20 2C 73 73 65 63 028A
63 77 20 64 65 6E 72 75 74 65 72 20 0296
3A 73 02A2
02A4 317
```

```
75 6D 69 78 61 4D 000002AC'010E0000' 02A4 318 TIME_OUT_ERR:
65 65 63 78 65 20 65 6D 69 74 20 6D 02A4 319 .ASCID /Maximum time exceeded. Current test section aborted./
74 6E 65 72 72 75 43 20 2E 64 65 64 02B2
6F 69 74 63 65 73 20 74 73 65 74 20 02BE
2E 64 65 74 72 6F 62 61 20 6E 02CA
02D6
02E0 320
02E0 321 NO_SET_TIM:
02E0 322 .ASCID /Maximum timer is already set, unable to change it./
02EE
02FA
0306
0312
031A 323
031A 324 FILE:
0326 .ASCID /file/ ; fills in RMS_ERR_STRING
0326 326
0326 327 RECORD:
0326 .ASCID /record/ ; fills in RMS_ERR_STRING
0334 328
0334 329 RMS_ERR_STRING:
0334 330 .ASCID /RMS !AS error in file !AD/ ; announces an RMS error
0342 331
034E
0355 332
0355 333 ZERO_PARCNT:
0363 334 .ASCID /PARCNT is less than, or equal to zero, PARCNT = 1 used./
036F
037B
0387
0393
0394 335
0394 336 PROC_ACT:
0394 337 .ASCID /Processes are currently active, unable to perform assignment./
03A2
03AE
03BA
03C6
03D2
03D9 338
03D9 339 NO_SET_LOG:
03E7 340 .ASCID /Unable to change log file name, assignment skipped./
03F3
03FF
040B
0414 341
```

TSTCNTRL
V04-000

TEST PACKAGE CONTROL PROGRAM
Read-Only Data

L 1

16-SEP-1984 01:30:05
5-SEP-1984 04:26:03

VAX/VMS Macro V04-00
[UETP.SRC]UETPHAS00.MAR;1

Page 9
(6)

TS
V0

	0414	343	CNTRLC_MSK:		
00000000	0414	344	.LONG 0		; mask which holds ASCII code for
00000008	0418	345	.LONG ^X0008		; the CNTRL-C
	041C	346			
	041C	347		; to hold extension for log files	
	041C	348			
	041C	349	LOG_EXT_DESC:		
47 4F 4C 2E 00000424'010E0000'	041C	350	.ASCID /.LOG/		; descriptor of extension for log fi
	0428	351			
	0428	352	NIL_EXT_DESC:		; nil descriptor so that MAKE
00000430'010E0000'	0428	353	.ASCID //		; TMP_LOG_NAM could be used to
	0430	354			; make a logical name
	0430	355			

```

0430 357 ; area to hold records to be written into the temporary command procedure
0430 358 ; which is created when parameters are to be passed to the command
0430 359 ; procedure being started up
0430 360
0430 361 COM_STAT_DESC: ; descriptor to get the final
0430 362 .ASCID /$ TCNTRLSTAT = $STATUS/ ; status of a command procedure
54 4E 43 54 20 24 00000438'010E0000' 0430
54 53 24 20 3D 20 54 41 54 53 4C 52 043E
53 55 54 41 044A
044E 363 ; which has parameters
044E 364
044E 365 COM_EXIT_DESC: ; descriptor to return the final
044E 366 .ASCID /$ EXIT TCNTRLSTAT/ ; status of a command procedure
54 49 58 45 20 24 00000456'010E0000' 044E
54 41 54 53 4C 52 54 4E 43 54 20 045C
0467 367 ; which has parameters
0467 368
0467 369 ; to hold extension for temporary command file
0467 370
0467 371 TMP_COM_EXT_DESC:
0467 372 .ASCID /.CM1/ ; descriptor of extension for
0473 373 ; command file with parameters
0473 374
0473 375 COM_IMAGE:
0473 376 .ASCID /SYS$SYSTEM:LOGINOUT.EXE/ ; image name for com file
59 53 24 53 59 53 0000047B'010E0000' 0473
55 4F 4E 49 47 4F 4C 3A 4D 45 54 53 0481
45 58 45 2E 54 048D

```

TSTCNTRL
V04-0C0

TEST PACKAGE CONTROL PROGRAM
Read-Only Data

N 1

16-SEP-1984 01:30:05 VAX/VMS Macro V04-00
5-SEP-1984 04:26:03 [UETP.SRC]UETPHAS00.MAR;1

Page 11
(8)

TS
V0

```
0492 378 ; READ-ONLY SECTION FOR PARSE
0492 379 ;
0492 380 ; FAO strings to hold the error messages for the parse routine.
0492 381 ; message consists of printing the record that failed and the token
0492 382 ; that it failed on.
0492 383
0492 384 RCRD_ERR_MSG: ; syntax error in record
0492 385 .ASCII \Syntax error with record. Record skipped -- !/ !AD \

78 61 74 6E 79 53 0000049A'010E0000' 04A0
20 68 74 69 77 20 72 6F 72 72 65 20 04AC
6F 63 65 52 20 2E 64 72 6F 63 65 72 04B8
2D 20 64 65 70 70 69 6B '3 20 64 72 04C4
20 44 41 21 20 2F 21 20 2D 04CD
04CD 386
04CD 387 TOKEN_ERR_MSG: ; token which failed
04CD 388 .ASCII \Invalid symbol or keyword. Symbol is -- !/ !AD \

69 6C 61 76 6E 49 000004D5'010E0000' 04CD
20 72 6F 20 6C 6F 62 6D 79 73 20 64 04DB
6D 79 53 20 2E 64 72 6F 77 79 65 68 04E7
2F 21 20 2D 2D 20 73 69 20 6C 6F 62 04F3
20 44 41 21 20 04FF
0504 389
0504 390
0504 391 ; ASCII strings of the various file types, used to determine what kind
0504 392 ; of file we have
0504 393
0504 394 EXE_STR:
0504 395 .ASCII /.EXE/ ; for images
0508 396
0508 397 COM_STR:
0508 398 .ASCII /.COM/ ; for command procedures

45 58 45 2E
4D 4F 43 2E
```

050C 400 .SBTTL Process Info Data Structure
00000000 401 .PSECT PROCINFO,WRT,NOEXE,PAGE
0000 402 :++

Now make a buffer to hold information regarding the detached processes.
When a process gets created it is given a slot. This slot holds information
regarding that process. The format of an individual slot is as follows:

offset	
0	FLINK
4	BLINK
8	FINALSTS
12	PID
16	TERMTIME
24	CPUTIM
28	PAGEFLTS
32	PGFLPEAK
36	WSPEAK
40	BIOCNT
44	DIOCNT
48	LOGIN
56	PRCINFO_FLGS
60	CHAN : T_ID
64	NODE ADDR
68	DEVICE ADDR
72	DIRECT ADDR
76	FILENAME ADDR
80	FIL_EXT ADDR
84	VERSION ADDR
88	NOD_SZ:SPC_SZ
92	DEV_SZ:DIR_SZ
96	FIL_SZ:EXT_SZ

```
0000 457 : 100      | VER_SZ: SPARE |
0000 458 :         | +-----+
0000 459 : 104      | |
0000 460 :         | |
0000 461 :         | | complete
0000 462 :         | |   file
0000 463 :         | | specification
0000 464 :         | |
0000 465 :         | |
0000 466 :         | |
0000 467 :         | +-----+
0000 468 :
0000 469 :
0000 470 : The structure is accessed by three pointers: the AVAIL_LIST_HEAD which points
0000 471 : to next available free slot, the COMPLETE_Q_HEAD which points to the head
0000 472 : of the queue of completed processes, and the CURRENT_LIST_HEAD which points
0000 473 : to a list of currently running processes. As slots are returned to the
0000 474 : available list, they are inserted at the head.
0000 475 :
0000 476 : --
0000 477 :
0000 478 : ; make enough initial space to hold NUMB_OF_SLOTS processes
0000 479 :
0000 480 PROC_INFO TOP: ; top of proc_info list
0000 481 .BLKB NUMB_OF_SLOTS * PROCINFO$ _PROC_INFO
0000 482
0000 483 ; now make headers for the different types of lists
0000 484
0000 485 ; first for the available slots list
0000 486
0000 487 AVAIL_LIST HEAD: ; header for available list
0000 488 .BLKB 1
0000 489
0000 490 ; for the completed process queue
0000 491
0000 492 COMPLETE_Q HEAD: ; header for completed
0000 493 .BLKB 1 ; process queue
0000 494
0000 495 ; and for the currently running processes list
0000 496
0000 497 CURRENT_LIST HEAD: ; header for currently running
0000 498 .BLKB 1 ; process list
0000 499
```

TSTCNTRL
V04-000

TEST PACKAGE CONTROL PROGRAM
Read/Write Data

D 2

16-SEP-1984 01:30:05 VAX/VMS Macro V04-00
5-SEP-1984 04:26:03 [UETP.SRC]UETPHAS00.MAR;1

Page 14
(10)

TS
VC

```

      4668 501      .SBTTL Read/Write Data
00000000 502      .PSECT RWDATA,WRT,NOEXE,PAGE
      0000 503
      0000 504
      0000 505      ; set up space to hold the values of the various counters
      0000 506
      0000 507 PROCESS_RUNNING:
00000000 0000 508      .LONG 0      ; # of active detached processes
      0004 509
      0004 510 TOTAL_PROC_RUN:
00000000 0004 511      .LONG 0      ; total # of detached processes
      0008 512
      0008 513 ACTIVE_PROC_LIMIT:
00000001 0008 514      .LONG 1      ; detached processes
      000C 515
      000C 516 TOTAL_PROC_LIMIT:
00000000 000C 517      .LONG 0      ; PARCNT, limit of total
      0010 518      ; detached processes
      0010 519 PRIV_MSK:
00000000 00000000 0010 520      .QUAD 0      ; quadword to hold privileges
```

```
0018 522 ;
0018 523 ; data area for inputs to the TSTCNTRL (FILNAM, DELLOG, REPORT,
0018 524 ; and PARCNT). there are descriptors of the keywords for the
0018 525 ; CLI and buffer areas to put the value associated with the keywords.
0018 526 ;
0018 527 ; also included here is the information needed to do a $GETDVI on
0018 528 ; SYSSCOMMAND. this is to find out if the TSTCNTRL is being run
0018 529 ; from a terminal or not (so that we know whether or not to
0018 530 ; enable control-c ast's.
0018 531 ;
0018 532 ;
0018 533 ; descriptors of keywords for the CLI
0018 534
0018 535 FAC_DESC: ; descriptor of FACILITY_CODE
49 4C 49 43 41 46 00000020'010E0000' 0018 536 .ASCID /FACILITY_CODE/
45 44 4F 43 5F 59 54 0026
002D 537
002D 538 FILNAM_DESC: ; descriptor of FILNAM
53 5F 45 4C 49 46 00000035'010E0000' 002D 539 .ASCID /FILE_SPEC/
43 45 50 003B
003E 540
003E 541
003E 542 PARCNT_DESC: ; descriptor of PARCNT
4C 4C 41 52 41 50 00000046'010E0000' 003E 543 .ASCID /PARALLEL_COUNT/
54 4E 55 4F 43 5F 4C 45 004C
0054 544
0054 545 REPORT_DESC: ; descriptor of REPORT
54 52 4F 50 45 52 0000005C'010E0000' 0054 546 .ASCID /REPORT_TYPE/
45 50 59 54 5F 0062
0067 547
0067 548 DELLOG_DESC: ; descriptor of DELLOG
45 54 45 4C 45 44 0000006F'010E0000' 0067 549 .ASCID /DELETE_LOG_FILE/
45 4C 49 46 5F 47 4F 4C 5F 0075
007E 550
007E 551 ; and descriptor areas for the respective values
007E 552
007E 553 FAC_IN: ; desc for FAC value
00000080 007E 554 .BLKW 1
02 0E 0080 555 .BYTE DSC$K_DTYPE_T,DSC$K_CLASS_D
00000086 0082 556 .BLKL 1
0086 557
0086 558 FILNAM_IN: ; desc for FILNAM value
00000088 0086 559 .BLKW 1
02 0E 0088 560 .BYTE DSC$K_DTYPE_T,DSC$K_CLASS_D
0000008E 008A 561 .BLKL 1
008E 562
008E 563 REPORT_IN: ; desc for REPORT value
00000090 008E 564 .BLKW 1
02 0E 0090 565 .BYTE DSC$K_DTYPE_T,DSC$K_CLASS_D
00000096 0092 566 .BLKL 1
0096 567
0096 568 PARCNT_IN: ; desc for PARCNT value
00000098 0096 569 .BLKW 1
02 0E 0098 570 .BYTE DSC$K_DTYPE_T,DSC$K_CLASS_D
0000009E 009A 571 .BLKL 1
009E 572
009E 573 ; the facility code that may have been changed
```

```
000000A2 009E 574
009E 575 FAC_CODE: ; new facility code
009E 576 .BLKL 1
00A2 577
00A2 578 ; item list for $GETDVI on SYSS$COMMAND
00A2 579
00A2 580 COMMAND_ITM_LST: ; item list for $GETDVI
0004 0004 00A2 581 .WORD 4,DVIS$ DEVCLASS ; find out if it is a terminal
00000106' 00A6 582 .LONG IN_DEVICE_CLASS ; put result here
00000000 00AA 583 .LONG 0
0020 0040 00AE 584 .WORD 64,DVIS$ DEVNAM ; get X-lated name of device
00000CC6' 00B2 585 .LONG IN_DEVICE_STR ; put name of device here
000000BE' 00B6 586 .LONG IN_DEVICE_DESC ; and size here
00000000 00BA 587 .LONG 0 ; end-of-item list
00BE 588
00000040 00BE 589 IN_DEVICE_DESC: ; descriptor to hold translated
000000C6' 00C2 590 .LONG 64 ; name of SYSS$COMMAND
00C6 591 .ADDRESS IN_DEVICE_STR
00C6 592
00000106 00C6 593 IN_DEVICE_STR: ; buffer to hold translated name
0106 594 .BLKB 64
0106 595
00000000 0106 596 IN_DEVICE_CLASS: ; buffer to hold value of class
0106 597 .LONG 0 ; characteristics
010A 598
010A 599 SYS_COMMAND: ; descriptor of SYSS$COMMAND
4F 43 24 53 59 53 00000112'010E0000' 010A 600 .ASCID /SYSS$COMMAND/
44 4E 41 4D 4D 0118
```

TSTCNTRL
V04-000

TEST PACKAGE CONTROL PROGRAM
Read/Write Data

G 2

16-SEP-1984 01:30:05
5-SEP-1984 04:26:03

VAX/VMS Macro V04-00
[UETP.SRC]UETPHAS00.MAR;1

Page 17
(12)

TS
VC

	011D	602	MBX_CHAN:	
0000	011D	603	.WORD 0	; channel # of termination mailbox
	011F	604		
	011F	605	TERM_BUFF:	
00000173	011F	606	.BLKB MAILBOX_SIZE	; buffer to hold mailbox
	0173	607		
	0173	608	LOG_FIL_DESC:	; desc. for perm. log file name
0000000C	0173	609	.LONG 12	
0000017B	0177	610	.ADDRESS LOG_FIL_STR	
	017B	611		
	017B	612	LOG_FIL_STR:	
47 4F 4C 2E 4C 52 54 4E 43 54 53 54	017B	613	.ASCII /TSTCNTRL.LOG/	; default name
00000286	0187	614	.BLKB NAMSC_MAXRSS	; room for more
	0286	615		
	0286	616	MSG_BLOCK:	
0000028A	0286	617	.BLKB 4	; hold \$GETMSG info
	028A	618		

```
028A 620 ; descriptors for start and stop sentinels
028A 621
028A 622 START_DESC:
00000005' 028A 623 .LONG START_STR_LEN ; fill in with default
00000292' 028E 624 .ADDRESS START_STR
0292 625
0292 626 START_STR:
4E 49 47 45 42 0292 627 .ASCII /BEGIN/
00000005 0297 628 START_STR_LEN = .-START_STR
0000031B 0297 629 .BLKB RCRD_SIZE ; extra space if needed
031B 630
031B 631
031B 632 STOP_DESC:
00000005' 031B 633 .LONG STOP_STR_LEN ; fill in with default
00000323' 031F 634 .ADDRESS STOP_STR
0323 635
0323 636 STOP_STR:
44 45 44 4E 45 0323 637 .ASCII /ENDED/
00000005 0328 638 STOP_STR_LEN = .-STOP_STR
000003AC 0328 639 .BLKB RCRD_SIZE ; extra space if needed
03AC 640
03AC 641
03AC 642 GETCHN_DESC:
00000074' 03AC 643 .LONG DIBSK_LENGTH
000003B4' 0380 644 .ADDRESS GETCHN_BUF
03B4 645
03B4 646
03B4 647 GETCHN_BUF:
00000428 03B4 648 .BLKB DIBSK_LENGTH ; buffer to hold get channel info
0428 649
0428 650
0428 651 MBX_UNIT:
0000 0428 652 .WORD 0 ; hold the mailbox unit number
```

```
042A 654 :++
042A 655 : This area is for the $CREPRC parameter list. It gets changed by the
042A 656 : Create Process Subroutine. The default values are the ones shown.
042A 657 :--
042A 658
042A 659 CREPRC_LIS:
00000000 042A 660 .LONG 12 ; number of parameters
042E 661 PIDADR:
00000000 042E 662 .LONG 0
0432 663 IMAGE:
00000000 0432 664 .LONG 0
0436 665 INPUT:
00000000 0436 666 .LONG 0
043A 667 OUTPUT:
00000000 043A 668 .LONG 0
043E 669 ERROR:
00000000 043E 670 .LONG 0
0442 671 PRVADR:
00000000 0442 672 .LONG 0
0446 673 QUOTA:
0000045E' 0446 674 .ADDRESS QUOTA_TABLE ; use same quota's as parent
044A 675 PRCNAM:
00000000 044A 676 .LONG 0
044E 677 BASPRI:
00000002 044E 678 .LONG 2
0452 679 UIC:
00000000 0452 680 .LONG 0
0456 681 MBXUNT:
00000000 0456 682 .LONG 0
045A 683 STSFLG:
00000000 045A 684 .LONG 0
```

```
045E 686 ; table to hold results of $GETJPI of the quota's of the parent process
045E 687 ; to be used for $CREPRC
045E 688
045E 689 QUOTA_TABLE:
01 045E 690 .BYTE PQL$ASTLM
00000006 045F 691 ASTLM: .LONG 6
02 0463 692 .BYTE PQL$BIOLM
00000006 0464 693 BIOLM: .LONG 6
03 0468 694 .BYTE PQL$BYTLM
00002000 0469 695 BYTLM: .LONG 8192
04 046D 696 .BYTE PQL$CPULM
00000000 046E 697 CPULM: .LONG 0
05 0472 698 .BYTE PQL$DIOLM
00000006 0473 699 DIOLM: .LONG 6
0C 0477 700 .BYTE PQL$ENQLM
00000032 0478 701 ENQLM: .LONG 50
06 047C 702 .BYTE PQL$FILLM
0000000A 047D 703 FILLM: .LONG 10
07 0481 704 .BYTE PQL$PGFLQUOTA
00000800 0482 705 PGFLQUOTA: .LONG 2048
08 0486 706 .BYTE PQL$PRCLM
00000008 0487 707 PRCLM: .LONG 8
09 048C 708 .BYTE PQL$TQELM
00000008 048D 709 TQELM: .LONG 8
0B 0490 710 .BYTE PQL$WSDEFAULT
00000064 0491 711 WSDEFAULT: .LONG 100
0A 0495 712 .BYTE PQL$WSQUOTA
00000078 0496 713 WSQUOTA: .LONG 120
00 049A 714 .BYTE PQL$LISTEND
```

TSTCNTRL
V04-000

TEST PACKAGE CONTROL PROGRAM
Read/Write Data

K 2

16-SEP-1984 01:30:05 VAX/VMS Macro V04-00
5-SEP-1984 04:26:03 [UETP.SRC]UETPHAS00.MAR;1

Page 21
(16)

```
049B 728 ; area to hold longwords for the flags
049B 729
049B 730 ; hold the main control case statement flags
049B 731
049B 732 CONTROL_CASE: ; case statement flags
00000000 049B 733 .LONG 0
049F 734
049F 735 ; hold the create process case statement flags
049F 736
049F 737 CREATE_PROC_CASE: ; create process case flags
00000000 049F 738 .LONG 0
04A3 739
04A3 740 ; hold the TSTCNTRL flags
04A3 741
04A3 742 FLAGS: ; TSTCNTRL flags
0000003E 04A3 743 .LONG ^X003E
04A7 744
04A7 745 ASSIGN_CASE: ; flags for the case for the
00000000 04A7 746 .LONG 0 ; assignment routines
```

```
00000000 00000000 04AB 748 PROC_TERM IOSB: ; IOSB to hold status of QIO for a terminated process
04AB 749 .QUAD 0
04B3 750
04B3 751 PRCNAMDESC: ; descriptor for user process name
00000000 04B3 752 .LONG 0
000004BB' 04B7 753 .ADDRESS PRCNAMSTR
04BB 754
04BB 755 PRCNAMSTR:
000004CA 04BB 756 .BLKB PROCESS_NAME_SIZ
04CA 757
04CA 758 EXIT_DESC: ; Exit handler descriptor
00000000 04CA 759 .LONG 0
0000186A' 04CE 760 .ADDRESS EXIT_HANDLER
00000001 04D2 761 .LONG 1
000004DA' 04D6 762 .ADDRESS STATUS
04DA 763
04DA 764 STATUS:
0000002C 04DA 765 .LONG $$$_ABORT ; status on exit
```

```
0004 04DE 767 GETUIC:
0304 04DE 768 .WORD 4 ; buffer for UIC from $GETJPI
000004FA' 04E0 769 .WORD JPI$ UIC ; code for UIC
00000000 04E2 770 .ADDRESS UIC_CODE ; put it in this address
000F 04E6 771 .LONG 0 ; not used here
031C 04EA 772 .WORD 15 ; size for process name
000004BB' 04EC 773 .WORD JPI$ PRCNAM ; get user process name
000004B3' 04EE 774 .ADDRESS PRCNAMSTR ; put it here
00000000 04F2 775 .ADDRESS PRCNAMDESC ; put size in descriptor
04F6 776 .LONG 0 ; terminate block
04FA 777
00000000 04FA 778 UIC_CODE:
04FA 779 .LONG 0 ; holds current UIC
04FE 780
04FE 781 LNMGRPNUM: ; Equivalence name giving the table...
5F 50 55 4F 52 47 24 4D 4E 4C 04FE 782 .ASCII /LNM$GROUP / ; ...name in which mbx logical names...
0000000A 0508 783 LNMGRP = .-LNMGRPNUM
20'20'20'20'20'20' 0508 784 .BYTE ^A/ /[6]
00000010 050E 785 LNMGRPLEN = .-LNMGRPNUM ; ...are to be put
050E 786
050E 787 LNMGRPNIL: ; Descriptor to convert...
0001 0006 050F 788 .WORD LNMGRPLEN-LNMGRP,DSC$K_CLASS_S
00000508' 0512 789 .ADDRESS LNMGRPNUM+LNMGRP
```

TSTCNTRL
V04-000

TEST PACKAGE CONTROL PROGRAM
Read/Write Data

N 2

16-SEP-1984 01:30:05
5-SEP-1984 04:26:03

VAX/VMS Macro V04-00
[UEIP.SRC]UEIPHAS00.MAR:1

Page 24
(20)

TS
VC

```

0516 791
0516 792 PRC_NAM_DESC:
00000008 0516 793 .LONG 8
0000051E' 051A 794 .ADDRESS PRC_NAM_STR
051E 795
051E 796 PRC_NAM_STR:
4C 52 54 4E 43 54 53 54 051E 797 .ASCII /TSTCNTRL/
0000052D 0526 798 .BLKB 7

```

```
; holds process name of TSTCNTRL
```

```
; default process name
; space for a different name
```

```
052D 800 ; for error messages
052D 801
052D 802 FAO_BUF: ; FAO output string descriptor
0000 0084 052D 803 .WORD RCRD_SIZE,0
0000053D 0531 804 .ADDRESS BUFFER
0535 805
0535 806 BUFFER_PTR: ; fake buffer for misc. strings
0000 0084 0535 807 .WORD RCRD_SIZE,0
0000053D 0539 808 .ADDRESS BUFFER
053D 809
000005C1 053D 810 BUFFER: ; FAO output buffer
053D 811 .BLKB RCRD_SIZE
05C1 812
0000 0084 05C1 813 GETMSG_DESC: ; descriptor for $GETMSG string
000005D1 05C5 814 .WORD RCRD_SIZE,0
05C9 815 .ADDRESS GETMSG_BUF
05C9 816
0000 0084 05C9 817 GETMSG_PTR:
000005D1 05CD 818 .WORD RCRD_SIZE,0
05D1 819 .ADDRESS GETMSG_BUF
05D1 820
00000655 05D1 821 GETMSG_BUF:
0655 822 .BLKB RCRD_SIZE
0655 823
00000000 0655 824 TTCHAN:
0655 825 .LONG 0 ; channel associated w/ contrl. term
```

```
0659 827 ; READ/WRITE SECTION FOR PARSE
0659 828 ;
0659 829 ; this section holds the results of the parse routine. The labels
0659 830 ; here are global throughout the entire TSTCNTRL and are used by the
0659 831 ; various routines.
0659 832
0000065D 0659 833 PARSE_FLAGS: ; flags which are used by the
0659 834 .BLKL 1 ; Parse routine
065D 835
00000084 065D 836 DATA_RCRD_DESC: ; descriptor of the input data
065D 837 .LONG RCRD_SIZE ; record. record is placed here
00000665 0661 838 .ADDRESS DATA_RCRD_STR ; by RMS
0665 839
000006E9 0665 840 DATA_RCRD_STR: ; input record goes here
0665 841 .BLKB RCRD_SIZE
06E9 842
00000084 06E9 843 COMMENT_DESC: ; descriptor to hold
06E9 844 .LONG RCRD_SIZE ; comment field
000006F1 06ED 845 .ADDRESS COMMENT_STR
06F1 846
00000775 06F1 847 COMMENT_STR: ; comment goes here
06F1 848 .BLKB RCRD_SIZE
0775 849
00000000 00000000 0775 850 SPEC_DESC: ; descriptor to hold the T-parsed
0775 851 .QUAD 0 ; filespec
077D 852
077D 853 ; the following all pertain to the resultant filespec. this includes
077D 854 ; the complete filespec and descriptors pointing to the different
077D 855 ; pieces of it.
077D 856
0000C0FF 077D 857 FILE_SPEC_DESC: ; descriptor of the resultant
077D 858 .LONG NAMSC_MAXRSS ; filespec after RMS
00000785 0781 859 .ADDRESS FILE_SPEC_STR ; $PARSE
0785 860
00000884 0785 861 FILE_SPEC_STR: ; the actual resultant filespec
0785 862 .BLKB NAMSC_MAXRSS
0884 863
0884 864 ; these are the descriptors pointing into the filespec
0884 865
0000088C 0884 866 NODE_DESC: ; point to the node part
0884 867 .BLKQ 1
088C 868
00000894 088C 869 DEVICE_DESC: ; point to the device part
088C 870 .BLKQ 1
0894 871
0000089C 0894 872 DIRECTORY_DESC: ; point to the directory part
0894 873 .BLKQ 1
089C 874
000008A4 089C 875 FILENAME_DESC: ; point to the filename part
089C 876 .BLKQ 1
08A4 877
0000C8AC 08A4 878 TYPE_DESC: ; point to the type field
08A4 879 .BLKQ 1
08AC 880
000008B4 08AC 881 VERSION_DESC: ; point to the version number
08AC 882 .BLKQ 1
08B4 883
```

```
08B4 884 ; now set up data area for the remaining parts which are the results of
08B4 885 ; the parse
08B4 886
00000084 08B4 887 ASSIGN_VALUE_DESC: ; descriptor to hold the value
000008BC 08B4 888 .LONG RCRD_SIZE ; of an assignment
08BC 889 .ADDRESS ASSIGN_VALUE_STR ; statement
08BC 890
0000C940 08BC 891 ASSIGN_VALUE_STR: ; the actual value
08BC 892 .BLKB RCRD_SIZE
0940 893
0940 894
00000084 0940 895 PARAM_DESC: ; descriptor to hold the
00000948 0940 896 .LONG RCRD_SIZE ; parameter string
0944 897 .ADDRESS PARAM_STR
0948 898
000009CC 0948 899 PARAM_STR: ; the actual string
0948 900 .BLKB RCRD_SIZE
09CC 901
000009CD 09CC 902 DELIMIT: ; 1 byte buffer to hold delimiter
09CC 903 .BLKB 1 ; of parameter string
09CD 904
00000000 00000000 09CD 905 TIME_DESC: ; descriptor the hold unconverted
09CD 906 .QUAD 0 ; time string
09D5 907
000009DD 09D5 908 TIME_VALUE: ; quadword to hold converted
09D5 909 .BLKQ 1 ; system time
09DD 910
09DD 911
09DD 912 ; make RMS structures to parse the filespec
09DD 913 .ALIGN LONG
09E0 914
09E0 915 PRSE_FILESPC_FAB: ; FAB for the parse
09E0 916 $FAB=
09E0 917 NAM = PRSE_FILESPC_NAM
0A30 918
0A30 919 PRSE_FILESPC_NAM: ; NAM to hold the filespec
0A30 920 $NAM=
0A30 921 ESA = FILE_SPEC_STR,- ; expanded filespec goes here
0A30 922 ESS = NAM$C_MAXRSS ; this is maximum size
0A90 923
```

```
0A90 925 .SBTTL RMS-32 Data Structures
0A90 926 .ALIGN LONG
0A90 927
0A90 928 DATA_FILE FAB:
0A90 929 $FAB - ; FAB for the data file
0A90 930 ORG = <SEQ>,- ; sequential file only
0A90 931 MRS = RCRD_SIZE,-
0A90 932 RFM = VAR ; allow variable records
0A90 933
0A90 934 DATA_FILE RAB:
0A90 935 $RAB - ; RAB for the data file records
0A90 936 FAB = DATA_FILE FAB,-
0A90 937 PBF = TCNTRL_PRMT,- ; address of prompt string
0A90 938 PSZ = TCNTRL_PRMT_SIZ,- ; size of prompt string
0A90 939 ROP = PMT,-
0A90 940 UBF = DATA_RCRD_STR,-
0A90 941 USZ = RCRD_SIZE
0B24 942
0B24 943 TMP_LOG FAB:
0B24 944 $FAB - ; FAB for temporary log files
0B24 945 ORG = <SEQ>,- ; FNA & FNS will be done dynamically
0B24 946 MRS = LARGE_BUFFER,-
0B24 947 RAT = <CR>,-
0B24 948 RFM = VAR
0B74 949
0B74 950 TMP_LOG RAB:
0B74 951 $RAB -
0B74 952 FAB = TMP_LOG FAB,-
0B74 953 USZ = LARGE_BUFFER
0B88 954
0B88 955 CNTRL_LOG FAB:
0B88 956 $FAB - ; FAB for perm log file
0B88 957 FNA = LOG_FIL_STR,- ; holds the name
0B88 958 FNS = 13,- ; holds the default size
0B88 959 ORG = <SEQ>,-
0B88 960 MRS = LARGE_BUFFER,-
0B88 961 RFM = VAR,-
0B88 962 RAT = <CR>
0C08 963
0C08 964 CNTRL_LOG RAB:
0C08 965 $RAB - ; RBF & RSZ fields will be
0C08 966 FAB = CNTRL_LOG_FAB ; inserted dynamically
0C4C 967
0C4C 968 COM_FAB:
0C4C 969 $FAB - ; FAB for temporary file used to
0C4C 970 ORG = <SEQ>,- ; start up .COM with parameters
0C4C 971 MRS = LARGE_BUFFER,-
0C4C 972 RAT = <CR>,-
0C4C 973 RFM = VAR
0C9C 974
0C9C 975 COM_RAB:
0C9C 976 $RAB - ; RAB for .COM file with parameters
0C9C 977 FAB = COM_FAB ; RBF and RSZ will be inserted dynam
0CE0 978
0CE0 979 DEL_COM FAB:
0CE0 980 $FAB - ; FAB to erase temp. command files
0CE0 981 ORG = <SEQ>,-
```

TS*CN1RL
V04-000

TEST PACKAGE CONTROL PROGRAM
RMS-32 Data Structures

F 3

16-SEP-1984 01:30:05
5-SEP-1984 04:26:03

VAX/VMS Macro V04-00
[UETP.SRC]UETPHAS00.MAR;1

Page 29
(23)

T
V

OCE0 982
OCE0 983
OD30 984

MRS = LARGE_BUFFER,-
RFM = VAR

```
0D30 986 .SBTTL Local area for ID descriptor
0D30 987 .PSECT ID_DESC_LOCAL,ABS,NOEXE,NOWRT,PAGE
0000 988
0000 989 ; PSECT to define offsets of local variables to the create process-
0000 990 ; name subroutine. No values are stored here, these are only
0000 991 ; offsets and the actual values are stored on the stack.
0000 992
0000 993 ID_DESC:
00000004 0000 994 .BLKL 1 ; to hold size of ID string
0004 995
00000008 0004 996 ID_ADDR:
0004 997 .BLKL 1 ; to hold address of string
0008 998
0000000C 0008 999 ID_STR:
0008 1000 .BLKB ID_STR_SIZ ; ID string
000C 1001
00000010 000C 1002 ID_NUMB:
000C 1003 .BLKL 1 ; ID number
0010 1004
00000010 0010 1005 ID_DESC_LENGTH = . ; length of block
0010 1006
```

```
0010 1008 .SBTTL Local area for Process Termination Subroutine
0010 1009 .PSECT PROC_TERM_LOCAL,ABS,NOEXE,NOWRT,PAGE
0000 1010
0000 1011 ; PSECT to define offsets of local variables for the Process Termination
0000 1012 ; Subroutine. No values are stored here, these are only offsets
0000 1013 ; and the actual values are stored on the stack.
0000 1014
0000 1015 TRMNTN_COM_DESC: ; descriptor to delete temporary
00000004 0000 1016 .BLKL 1 ; command file
00000008 0004 1017 .BLKL 1 ; address of string
0008 1018
0008 1019 TRMNTN_COM_STR:
00000015 0008 1020 .BLKB FILENAME_LENGTH+FILE_EXT_LENGTH ; temporary command file name
0015 1021
0015 1022 LOG_RCRD_DESC: ; descriptor to hold temporary
00000019 0015 1023 .BLKL 1 ; log file record
0000001D 0019 1024 .BLKL 1 ; address of record
001D 1025
001D 1026 LOG_RCRD_STR: ; log file record
0000011D 001D 1027 .BLKB LARGE_BUFFER
011D 1028
011D 1029 TEMP_BUFF_DESC: ; temp. buffer to hold capitalized
00000121 011D 1030 .BLKL 1 ; record to check for sentinels
00000125 0121 1031 .BLKL 1 ; address of string
0125 1032
0125 1033 TEMP_BUFF_STR: ; capitalized string
00000225 0125 1034 .BLKB LARGE_BUFFER
0225 1035
0225 1036 TRMNTN_FIL_DESC: ; descriptor for terminated
00000229 0225 1037 .BLKL 1 ; file name
0000022D 0229 1038 .BLKL 1 ; pointer to filename
022D 1039
022D 1040 TRMNTN_LOG_DESC: ; descriptor for temp. log file
00000231 022D 1041 .BLKL 1 ; name
00000235 0231 1042 .BLKL 1 ; address of log file name
0235 1043
0235 1044 TRMNTN_LOG_STR: ; terminated log file
00000242 0235 1045 .BLKB FILENAME_LENGTH+FILE_EXT_LENGTH
0242 1046
0242 1047 RTN_NAM_DESC: ; descriptor for the returned
00000246 0242 1048 .BLKL 1 ; process' process name
0000024A 0246 1049 .BLKL 1 ; address of process name
024A 1050
024A 1051 RTN_NAM_STR: ; returned process name
00000259 024A 1052 .BLKB PROCESS_NAME_SIZE
0259 1053
00000259 0259 1054 PROC_TERM_LENGTH = . ; length of this block
0259 1055
```

```
0259 1057 .SBTTL Local area for create command procedure w/ parameters
0259 1058 .PSECT TEMP_COM_LOCAL,ABS,NOEXE,NOWRT,PAGE
0000 1059
0000 1060 ; PSECT to define offsets of local variables for creating a command
0000 1061 ; procedure to which parameters are to be passed. No values are
0000 1062 ; stored here, these are only the offsets and the actual values
0000 1063 ; are stored on the stack.
0000 1064
0000 1065 TEMP_COM_DESC: ; holds the filename for the
0000 1066 .BLKL 1 ; temporary command file (used
00000004 0000 1067 .BLKL 1 ; when passing parameters)
00000008 0008 1068
0008 1069 TEMP_COM_STR: ; holds the filename
00000015 0008 1070 .BLKB FILENAME_LNGTH+FILE_EXT_LNGTH
0015 1071
0015 1072 COM_LINE_DESC: ; holds the command line to start
00000019 0015 1073 .BLKL 1 ; up the real procedure. done
0000001D 0019 1074 .BLKL 1 ; as an indirect procedure
001D 1075
001D 1076 COM_LINE_STR: ; the complete filespec (look like
0000011D 001D 1077 .BLKB LARGE_BUFFER ; '$@filespec -')
011D 1078
000U011D 011D 1079 COM_PARM_LENGTH = . ; length of this block
011D 1080
```

```
011D 1082 .SBTTL Local area for Create Process Subroutine
011D 1083 .PSECT CREAT_PROC_LOCAL,ABS,NOEXE,NOWRT,PAGE
0000 1084
0000 1085 ; PSECT to define offsets of local variables for the Create Process
0000 1086 ; Subroutine. No values are stored here, these are only offsets
0000 1087 ; and the actual values are stored on the stack.
0000 1088
0000 1089 TEMP_PNAM_DESC: ; holds the process name of
00000004 0000 1090 .BLKL 1 ; the detached process
00000008 C004 1091 .BLKL 1
0008 1092
0008 1093 TEMP_PNAM_STR: ; holds the actual process name
00000017 0008 1094 .BLKB PROCESS_NAME_SIZE
0017 1095
0017 1096 TEMP_LOG_DESC: ; holds the log filename of the
0000001B 0017 1097 .BLKL 1 ; detached process
0000001F 001B 1098 .BLKL 1
001F 1099
001F 1100 TEMP_LOG_STR: ; holds the actual log filename
0000002C 001F 1101 .BLKB FILENAME_LENGTH+FILE_EXT_LENGTH
002C 1102
002C 1103 MBX_LOGNAM_DESC: ; holds the logical name of a
00000034 002C 1104 .BLKB 1 ; mailbox which holds the
0034 1105 ; parameters to an image
0034 1106
0034 1107 MBX_LOGNAM_STR: ; logical name goes here
0000003D 0034 1108 .BLKB FILENAME_LENGTH
003D 1109
003D 1110 STRCT_TOP_BOT: ; 1st longword holds top of new
00000045 003D 1111 .BLKB 1 ; structure, 2nd holds bottom
0045 1112
00000045 0045 1113 CREATE_PROC_LENGTH = . ; length of this block
0045 1114
```

```
0045 1116 .SBTTL Local area for TPARSE and Get/Parse Routine
0045 1117 .PSECT GET_PARSE_LOCAL,ABS,NOEXE,NOWRT,PAGE
0000 1118
0000 1119 ; PSECT to define offsets of local variables for the TPARSE routine
0000 1120 ; and for getting a record and parsing that record. No values are
0000 1121 ; stored here, these are only offsets and the actual values are stored
0000 1122 ; on the stack.
0000 1123
0000 1124 DAT_PARSE_BLK: ; parameter block for TPARSE
0000 1125 .BLKL 1 ; count field (initialized to 8)
00000004 0000 1125 .BLKL 1 ; flag field
00000008 0004 1126 .BLKB TPASK_LENGTH0-8 ; buffer for rest of block
00000024 0008 1127
0024 1128
0024 1129 RCRD_ERR_DESC: ; holds the syntax error message
0000002C 0024 1130 .BLKQ 1 ; and the invalid record
002C 1131
002C 1132 RCRD_ERR_FAO: ; descriptor for input to the
00000034 002C 1133 .BLKQ 1 ; FAO service
0034 1134
0034 1135 RCRD_ERR_STR: ; error message goes here
0000013C 0034 1136 .BLKB 2*RCRD_SIZE
013C 1137
013C 1138 TOKEN_ERR_DESC: ; holds the invalid token message
00000144 013C 1139 .BLKQ 1 ; and the token
0144 1140
0144 1141 TOKEN_ERR_FAO: ; descriptor for input to the
0000014C 0144 1142 .BLKQ 1 ; FAO service
014C 1143
014C 1144 TOKEN_ERR_STR: ; message goes here
00000254 014C 1145 .BLKB 2*RCRD_SIZE
0254 1146
00000254 0254 1147 GET_PARSE_LENGTH = . ; length of this block
0254 1148
```

```
0254 1150 .SBTTL Main Program
0254 1151 :++
0254 1152 : FUNCTIONAL DESCRIPTION:
0254 1153 :
0254 1154 : This is the main routine of the TSTCNTRL, it will determine what
0254 1155 : the input record is and take appropriate action according to what
0254 1156 : the record is.
0254 1157 :
0254 1158 : CALLING SEQUENCE:
0254 1159 :
0254 1160 : None
0254 1161 :
0254 1162 : INPUT PARAMETERS:
0254 1163 :
0254 1164 : The logical names: FILNAM, PARCNT, DELLOG, and REPORT. The ASCII
0254 1165 : data file -- FILNAM.
0254 1166 :
0254 1167 : IMPLICIT INPUTS:
0254 1168 :
0254 1169 : None
0254 1170 :
0254 1171 : OUTPUT PARAMETERS:
0254 1172 :
0254 1173 : None
0254 1174 :
0254 1175 : IMPLICIT OUTPUTS:
0254 1176 :
0254 1177 : None
0254 1178 :
0254 1179 : COMPLETION CODES:
0254 1180 :
0254 1181 : The final status of the TSTCNTRL is sent to EXIT_HANDLER by the
0254 1182 : longword STATUS.
0254 1183 :
0254 1184 : SIDE EFFECTS:
0254 1185 :
0254 1186 : Leaves a log file called TSTCNTRL.LOG, may leave other log files
0254 1187 : if DELLOG is false.
0254 1188 :
0254 1189 :--
00000000 1190 .PSECT TSTCNTRL,EXE,NOWRT,PAGE
0000 1191
0000 1192 .DEFAULT DISPLACEMENT,WORD
0000 1193
0000 1194 .ENTRY TSTCNTRL,^M<> ; Entry mask
0002 1195
0002 1196 MOVAL SSERROR,(FP) ; declare an exception handler
0007 1197
0007 1198 $SETSFM_S ENBFLG = #1 ; Enable system service failure mode
0010 1199 $DCLEXH_S DESBLK = EXIT_DESC ; Declare an exit handler
001B 1200
001B 1201 ; get the facility code for output messages
001B 1202
001B 1203 PUSHAQ FAC_IN ; descriptor for fac. code
001F 1204 PUSHAQ FAC_DESC ; keyword FACILITY_CODE
0023 1205 CALLS #2,G^CLISGET_VALUE ; get the new code
002A 1206
```

```
002A 1207      ; convert the code to binary
002A 1208
009E'CF DF 002A 1209      PUSHAL FAC_CODE      ; binary code goes here
007E'CF 7F 002E 1210      PUSHAQ FAC_IN      ; ascii code number
00000000'GF 02 FB 0032 1211      CALLS #2,G^OTSS$CVT_TI_L      ; convert the number
0039 1212
0039 1213      ; get privileges
0039 1214
0010'CF 01 10 01 FO 0039 1215      INSV #1,#PRVSV_WORLD,#1,PRIV_MSK      ; need world in order to stop proces
0010'CF 01 05 01 FO 0040 1216      INSV #1,#PRVSV_DETACH,#1,PRIV_MSK      ; set mask for detach privilege
0047 1217      $SETPRV_S ENBFLG = #1,-      ; Get the detached privilege
0047 1218      PRVADR = PRIV_MSK
0058 1219
0058 1220      CHK_CNTL_C:
0058 1221      ; look at SYSS$COMMAND and see if it is coming from a terminal.
0058 1222      ; if it is then get the channel number and enable control-c's
0058 1223      ; to that terminal.
0058 1224
0058 1225      $GETDVIW_S DEVNAM = SYS_COMMAND,-      ; look at SYSS$COMMAND
0058 1226      EFN = #1,-      ; wait for this event flag
0058 1227      ITMLST = COMMAND_ITM_LST      ; get class and name
0072 1228
0072 1229      ; now, check the device class and see if it is a terminal
0072 1230
00000042 8F 0106'CF D1 0072 1231      CMPL IN_DEVICE_CLASS,#DC$_TERM      ; is device a terminal?
007B 1232      BNEQ 10$      ; no, don't enable control-c's
007D 1233
007D 1234      ; device is a terminal, so enable control-c's
007D 1235
007D 1236      $ASSIGN_S DEVNAM = IN_DEVICE_DESC,-      ; assign a channel to terminal
007D 1237      CHAN = TTCHAN
008E 1238
008E 1239      $QIOW_S CHAN = TTCHAN,-      ; with this channel,
008E 1240      FUNC = #IO$_SETMODE!IO$_CTRLCAST,-      ; establish a control-c
008E 1241      P1 = CCASTHAND      ; handler here.
00AF 1242
00AF 1243
00AF 1244      10$:
00AF 1245      ; get the data filename
00AF 1246
0086'CF 7F 00AF 1247      PUSHAQ FILNAM_IN      ; descriptor for FILNAM
002D'CF 7F 00B3 1248      PUSHAQ FILNAM_DESC      ; keyword for CLI
00000000'GF 02 FB 00B7 1249      CALLS #2,G^CCI$GET_VALUE      ; get the filnam
00BE 1250
00BE 1251      ; store filename in the FAB and OPEN it
00BE 1252
00BE 1253      $FAB_STORE FAB = DATA_FILE_FAB,-      ; the data fab
00BE 1254      FNA = @<DSC$A_POINTER+FILNAM_IN>,-      ; address of filename
00BE 1255      FNS = <DSC$_LENGTH+FILNAM_IN> ; size of filename
00CF 1256
00CF 1257      PEN_FILE:
00CF 1258
00CF 1259      ; now open the data file. File may be a PPF viz SYSS$INPUT
00CF 1260
00CF 1261      $OPEN FAB = DATA_FILE_FAB      ; open the data file
00DA 1262
00DA 1263      BLBS R0,10$      ; if successful, then continue
```

```
00DD 1264
00DD 1265 ; OPEN failed, then something wrong with FILNAM
00DD 1266 ; write error for invalid FILNAM
00DD 1267
018F'CF 7F 00DD 1268 PUSHAQ FILNAM_ERR ; error message
01 DD 00E1 1269 PUSHL #1
00C31134 8F DD 00E3 1270 PUSHL #TCNTRL$ TEXT!STSSK_SEVERE ; fatal error
00000000'GF 03 FB 00E9 1271 CALLS #3,G^LIB$SIGNAL ; write message
00F0 1272
0301 31 00F0 1273 BRW END_TCNTRL ; and exit
00F3 1274
00F3 1275 10$:
00F3 1276 ; connect the data stream
00F3 1277
00F3 1278 $CONNECT RAB = DATA_FILE_RAB,- ; connect the stream
00F3 1279 ERR = RMSEERR
0102 1280
0102 1281 GET_QUAL:
0102 1282 ; now get the other qualifiers from the CLI
0102 1283
0102 1284 ; the REPORT format
0102 1285
008E'CF 7F 0102 1286 PUSHAQ REPORT_IN ; descriptor for the value
0054'CF 7F 0106 1287 PUSHAQ REPORT_DESC ; descriptor of keyword for CLI
000C0000'GF 02 FB 010A 1288 CALLS #2,G^CIS$GET_VALUE ; get the value
0111 1289
0092'DF 91 0111 1290 CMPB @<DSC$A_POINTER+REPORT_IN>,- ; is report short?
53 8F 0115 1291 #^A/S/
0117 1292
04A3'CF 07 12 0117 1293 BNEQ 10$ ; no, check again
08 CA 0119 1294 BICL2 #TCNTRL$M_LONG_REPORT,FLAGS ; yes, change flag
1B 11 011E 1295 BRB 20$ ; continue
0120 1296
0120 1297 10$:
0092'DF 91 0120 1298 CMPB @<DSC$A_POINTER+REPORT_IN>,- ; is report long?
4C 8F 0124 1299 #^A/L/
13 13 0126 1300 BEQL 20$ ; yes, change nothing
0128 1301
0128 1302 ; no, so print warning and use default
0128 1303
01C4'CF DF 0128 1304 PUSHAL REPORT_ERR
01 DD 012C 1305 PUSHL #1
00C31130 8F DD 012E 1306 PUSHL #TCNTRL$ TEXT!STSSK_WARNING
00000000'GF 03 FB 0134 1307 CALLS #3,G^LIB$SIGNAL
013B 1308
013B 1309 20$:
013B 1310 ; get the concurrent process count (PARCNT)
013B 1311
0096'CF 7F 013B 1312 PUSHAQ PARCNT_IN ; descriptor for PARCNT
003E'CF 7F 013F 1313 PUSHAQ PARCNT_DESC ; keyword name for CLI
00000000'GF 02 FB 0143 1314 CALLS #2,G^CIS$GET_VALUE ; get the value
014A 1315
014A 1316 ; convert PARCNT string into a number
014A 1317
000C'CF DF 014A 1318 PUSHAL TOTAL_PROC_LIMIT ; place for converted number
0096'CF DF 014E 1319 PUSHAL PARCNT_IN ; string to be converted
00000000'GF 02 FB 0152 1320 CALLS #2,G^OTSS$CVT_TI_L ; convert text to longword
```

```
0159 1321
0159 1322
0159 1323
00 000C'CF D1 0159 1324
      18 14 015E 1325
      0355'CF DF 0160 1326
      01 DD 0164 1327
      00C31130 8F DD 0166 1328
00000000'GF 03 FB 016C 1329
      0173 1330
      000C'CF 01 D0 0173 1331
      0178 1332
      0178 1333 30$:
      0178 1334
      000C'CF D0 0178 1335
      0008'CF 017C 1336
      017F 1337
      017F 1338
      017F 1339
      0067'CF 7F 017F 1340
00000000'GF 01 FB 0183 1341
      018A 1342
50 00000000'8F D1 018A 1343
      05 12 0191 1344
      0193 1345
      04A3'CF 04 CA 0193 1346
      0198 1347
      0198 1348 40$:
      0198 1349
      0198 1350
      0198 1351
      0198 1352
      01AF 1353
      01AF 1354
      01AF 1355
      01AF 1356
      01AF 1357
      01AF 1358
      01AF 1359
      01DA 1360
      01DA 1361
      01DA 1362
      01DA 1363
      01DA 1364
      52 0384'CF DE 01F0 1365
0428'CF 0C A2 B0 01F5 1366
      01FB 1367
      01FB 1368
      01FB 1369
      4650'CF 7F 01FB 1370
      0000'CF DF 01FF 1371
      00000168 8F DD 0203 1372
      32 DD 0209 1373
      134C'CF 04 FB 020B 1374
      0210 1375
      0210 1376
      0210 1377

; see if valid PARCNT
CMPL TOTAL_PROC_LIMIT,#0 ; is PARCNT <= 0?
BGTR 30$ ; no, continue
PUSHAL ZERO_PARCNT ; yes, print warning .
PUSHL #1
PUSHL #TCNTRL$TEXT!ST$K_WARNING
CALLS #3,G^LIB$SIGNAL

MOVL #1,TOTAL_PROC_LIMIT ; . . . and use PARCNT = 1

MOVL TOTAL_PROC_LIMIT,- ; initialize maximum number of
      ACTIVE_PROC_LIMIT ; active processes

; lastly check to see if the temp. log files are to be deleted
PUSHAQ DELLOG_DESC ; keyword for CLI
CALLS #1,G^C[IS$PRESENT ; is the keyword there?

CMPL #CLIS_NEGATED,R0 ; was the value negated?
BNEQ 40$ ; no, use default

BICL2 #TCNTRL$M_DELETE_TEMP_LOG,FLAGS ; yes, change flag

; make connection for termination mailbox
$CREMBX_S CHAN = MBX_CHAN,- ; create termination mailbox
      MAXMSG = #MAILBOX_SIZE ; message from process

$QIO_S CHAN = MBX_CHAN,- ; do a read to mailbox
      FUNC = #IOS$ READVBLK,-
      ASTADR = PROC_TERM_AST,- ; on read, goto process termination
      IOSB = PROC_TERM_IOSB,-
      P1 = TERM_BOFF,-
      P2 = #MAILBOX_SIZE

; get unit number of mailbox for create process
$GETCHN_S CHAN = MBX_CHAN,- ; want unit # of mailbox
      PRIBUF = GETCHN_DESC
MOVAL GETCHN_BUF,R2 ; get starting addr of getchan
MOVW DIB$W_UNIT(R2),MBX_UNIT ; get the unit #

; initialize the data structure
PUSHAQ AVAIL_LIST_HEAD ; the avail. list pointer
PUSHAL PROC_INFO_TOP ; the addr of the top of struct.
PUSHL #PRCINFO$PRC_INFO ; the byte count of the slot size
PUSHL #NUMB_OF_SLOTS ; the number of slots we have
CALLS #4,INIT_STRUCT ; initialize the structure

; now get the UIC of the TSTCNTRL program
```

```
0210 1378 $GETJPIW_S ITMLST = GETUIC,- ; all we want is the UIC
0210 1379 EFN = #1
0225 1380
0225 1381 ; ensure that processes we create can access their SYSS$INPUT
0225 1382
0225 1383 PUSHL #2 ; we'll convert a 2-byte...
0227 1384 MOVZWL LNMGRPNIL,-(SP) ; ...integer to ASCII...
022C 1385 PUSHAL LNMGRPNIL
0230 1386 PUSHAL UIC CODE+2 ; ...using group number of...
0234 1387 CALLS #4,GETSS$CVT L TO ; ...the process which runs us
023B 1388 $CRELNM_S TABNAM = LNMPCDIR,- ; force SYSS$INPUT mbx name...
023B 1389 LOGNAM = LNMTMPMBX,- ; ...to appear in...
023B 1390 ITMLST = LNMITMLST ; ...a group logical name table
0252 1391
0252 1392 ; now get the quotas and base priority of the parent process
0252 1393
0252 1394 $GETJPIW_S ITMLST = QUOTA_LIS,- ; all quotas and priority
0252 1395 EFN = #1
0267 1396
0267 1397 GET_DATA_RCRD:
0267 1398
0267 1399 ; the first two records may be special assignment statements
0267 1400 ; get them and find out.
0267 1401
0267 1402 CLRL CONTROL_CASE ; make sure nothing is set
026B 1403
026B 1404 CALLS #0,GET_PARSE_SBRN ; get and parse the record
0270 1405
0270 1406 ; if it is an assignment statement, then go do it
0270 1407 ; otherwise, go directly to the begin message and don't get second record
0270 1408
0270 1409 BBC #CASE$V_ASSIGN,CONTROL_CASE,STRT_MSG ; it isn't assignment,
0276 1410 ; so goto start message
0276 1411
0276 1412 PUSHAL CONTROL_CASE
027A 1413 CALLS #1,ASSIGN_SBRN ; perform the assignment
027F 1414
027F 1415
027F 1416 ; we had one assignment, see if we have a second one
027F 1417
027F 1418 CALLS #0,GET_PARSE_SBRN ; get and parse the record
0284 1419
0284 1420 ; if it is an assignment statement, then go do it
0284 1421 ; otherwise, go directly to the begin message
0284 1422
0284 1423 BBC #CASE$V_ASSIGN,CONTROL_CASE,STRT_MSG ; it isn't assignment,
028A 1424 ; so goto start message
028A 1425
028A 1426 PUSHAL CONTROL_CASE
028E 1427 CALLS #1,ASSIGN_SBRN ; perform the assignment
0293 1428
0293 1429
0293 1430 STRT_MSG:
0293 1431 ; now, create the permanent log file
0293 1432
0293 1433 BISL2 #TCNTRL$M_SET_LOGNAM,FLAGS ; cannot change the logfile name
029C 1434
```

02 DD 0225 1383
7E 050E'CF 3C 0227 1384
050E'CF DF 022C 1385
04FC'CF DF 0230 1386
00000000'GF 04 FB 0234 1387
023B 1388
023B 1389
023B 1390
0252 1391
0252 1392
0252 1393
0252 1394
0252 1395
0267 1396
0267 1397
0267 1398
0267 1399
0267 1400
0267 1401
049B'CF D4 0267 1402
107D'CF 00 FB 026B 1403
026B 1404
0270 1405
0270 1406
0270 1407
0270 1408
1D 049B'CF 05 E1 0270 1409
0276 1410
0276 1411
049B'CF DF 0276 1412
0EAO'CF 01 FB 027A 1413
027F 1414
027F 1415
027F 1416
027F 1417
107D'CF 00 FB 027F 1418
0284 1419
0284 1420
0284 1421
0284 1422
09 049B'CF 05 E1 0284 1423
028A 1424
028A 1425
049B'CF DF 028A 1426
0EAO'CF 01 FB 028E 1427
0293 1428
0293 1429
0293 1430
0293 1431
0293 1432
04A3'CF 000000EC 8F C8 0293 1433
029C 1434

```
029C 1435 $CREATE FAB = CNTRL_LOG_FAB,- ; make the file
029C 1436 ERR = RMSERR
02AB 1437
02AB 1438 $CONNECT RAB = CNTRL_LOG_RAB,- ; establish a stream
02AB 1439 ERR = RMSERR
02BA 1440
02BA 1441 ; set the process name of the controlling process
02BA 1442
02BA 1443 $$SETPRN_S PRCNAM = PRC_NAM_DESC ; set default of process name
02C5 1444
02C5 1445 ; now print the starting time descriptor
02C5 1446
04A3'CF 20 C8 02C5 1447 BISL2 #TCNTRL$M_WRT_MSG,FLAGS ; always print this message
02CA 1448
02CA 1449 CLRL -(SP)
0516'CF 7E D4 02CA 1449 PUSHAL PRC_NAM_DESC
02 02CC 1450
00C3103B 02 DD 02D0 1451
00000000'GF 04 FB 02D2 1452
02D8 1453
02DF 1454
02DF 1455 ; now set the PARSE bit and start the main case loop
02DF 1456
049B'CF 00000080 8F C8 02DF 1457 BISL2 #CASE$M_PARSE,CONTROL_CASE ; make sure we do a parse
02E8 1458
02E8 1459 MAIN_LOOP:
02E8 1460 ; This is the loop which holds all the control for the various
02E8 1461 ; subroutines. Control goes to each subroutine depending on which
02E8 1462 ; bits in CONTROL_CASE are set.
02E8 1463
02E8 1464 ; determine if we are to continue with loop. use the following . . .
02E8 1465 ; do while ~TCNTRL$V_EXIT v PROCESS_CNT > 0
02E8 1466
02 04A3'CF 0B E0 02E8 1467 BBS #TCNTRL$V_EXIT,FLAGS,10$ ; is the exit flag set . . .
0A 11 02EE 1468 BRB 20$ ; no, so continue
02F0 1469
02F0 1470 10$:
00 0000'CF D1 02F0 1471 CMPL PROCESS_RUNNING,#0 ; . . . is the number of running
03 14 02F5 1472 BGTR 20$ ; processes <= 0?
00E0 31 02F7 1473 BRW END_MAIN ; yes, get out of loop
02FA 1474
02FA 1475 20$:
02FA 1476 ; now find out where we are suppose to case to
02FA 1477
02FA 1478 ; find the first set bit in the case word and then case off of it.
00 EA 02FA 1479 FFS #0,- ; start looking from position zero
049B'CF 20 02FC 1480 #<CASE$S_CNTRL_CASE*8>,- ; the size is this many bits
5A 02FD 1481 CONTROL_CASE,- ; this is the base
0300 1482 R10 ; and the results goes here
0301 1483
07 00 5A 8F 0301 1484 CASEB R10,#0,#<CASE$K_CNTRL_CASE_SIZE - 1> ; perform case
0305 1485 30$:
001A' 0305 1486 .WORD ABORT_TCNTRL_BLK - 30$
002A' 0307 1487 .WORD TIMER_EXP_BLK - 30$
003E' 0309 1488 .WORD PROCESS_TERM_BLK - 30$
0062' 030B 1489 .WORD EOF_BLK - 30$
0076' 030D 1490 .WORD FILE_BLK - 30$
0092' 030F 1491 .WORD ASSIGN_BLK - 30$
```

```
009E' 0311 1492 .WORD CREATE_PROC_BLK - 30$
00CA' 0313 1493 .WORD GET_PARSE_BLK - 30$
      0315 1494
      0315 1495 HIBER_BLK:
      0315 1496 ; reached here by falling through case with no match
      0315 1497 ; go into HIBER and sleep.
      0315 1498
      0315 1499 $HIBER_S
      031C 1500
      031C 1501 ; and do next case
      031C 1502
00B8 31 031C 1503 BRW CONTIN_MAIN ; perform next case
      031F 1504
      031F 1505
      031F 1506 ABORT_TCNTL_BLK:
      031F 1507 ; reached this via an AST setting CASE$V_ABORT
      031F 1508 ; begin abnormal abortion of the TSTCNTRL
      031F 1509
049B'CF DF 031F 1510 PUSHAL CONTROL_CASE ; flags for control case statement
04A3'CF DF 0323 1511 PUSHAL FLAGS ; flags for TSTCNTRL
0467'CF 02 FB 0327 1512 CALLS #2,ABORT_TCNTL_SBRTN ; start termination
      032C 1513
      032C 1514 ; do next case
      032C 1515
00A8 31 032C 1516 BRW CONTIN_MAIN ; perform next case
      032F 1517
      032F 1518 TIMER_EXP_BLK:
      032F 1519 ; reached this via an AST setting CASE$V_TIME_EXP
      032F 1520 ; stop currently running processes and continue
      032F 1521
0659'CF DF 032F 1522 PUSHAL PARSE_FLAGS ; flags saying what has been parsed
049B'CF DF 0333 1523 PUSHAL CONTROL_CASE ; flags for control case statement
04A3'CF DF 0337 1524 PUSHAL FLAGS ; flags for TSTCNTRL
04A3'CF 02 FB 033B 1525 CALLS #2,TIME_EXP_SBRTN ; stop processes
      0340 1526
      0340 1527 ; do next case
      0340 1528
0094 31 0340 1529 BRW CONTIN_MAIN ; perform next case
      0343 1530
      0343 1531 PROCESS_TERM_BLK:
      0343 1532 ; reached this via an AST setting CASE$V_PROC_TERM
      0343 1533 ; handle the terminated process
      0343 1534
04650'CF 7F 0343 1535 PUSHAQ AVAIL_LIST_HEAD ; header of available slots
031B'CF 7F 0347 1536 PUSHAQ STOP_DESC ; descriptor of ending sentinel
028A'CF 7F 034B 1537 PUSHAQ START_DESC ; descriptor of beginning sentinel
049B'CF DF 034F 1538 PUSHAL CONTROL_CASE ; flags for control case statement
04A3'CF DF 0353 1539 PUSHAL FLAGS ; flags for TSTCNTRL
0000'CF DF 0357 1540 PUSHAL PROCESS_RUNNING ; number of processes still running
04658'CF 7F 035B 1541 PUSHAQ COMPLETE_Q_HEAD ; header of completed process queue
0506'CF 07 FB 035F 1542 CALLS #7,PROCESS_TERM_SBRTN ; handle the terminated process
      0364 1543
      0364 1544 ; and do next case
      0364 1545
0070 31 0364 1546 BRW CONTIN_MAIN ; perform next case
      0367 1547
      0367 1548
```

```
0367 1549 EOF_BLK:
0367 1550 ; reached this by GET_PARSE setting CASE$V_EOF
0367 1551 ; handle the end-of-file
0367 1552
0AEO'CF DF 0367 1553 PUSHAL DATA_FILE_RAB ; start of data file RAB
04A3'CF DF 036B 1554 PUSHAL FLAGS ; flags for the TSTCNTRL
049B'CF DF 036F 1555 PUSHAL CONTROL_CASE ; flags for control case statement
08C3'CF 03 FB 0373 1556 CALLS #3,EOF_SBRTN ; handle end-of-file
0378 1557
0378 1558 ; and do next case
0378 1559
005C 31 0378 1560 BRW CONTIN_MAIN ; perform next case
037B 1561
037B 1562 FILE_BLK:
037B 1563 ; reached this by GET_PARSE or PROC_TERM setting CASE$V_FILE
037B 1564 ; handle having a file spec
037B 1565
0008'CF DF 037B 1566 PUSHAL ACTIVE_PROC_LIMIT ; max process active
0000'CF DF 037F 1567 PUSHAL PROCESS_RUNNING ; number processes currently active
0659'CF DF 0383 1568 PUSHAL PARSE_FLAGS ; flags set by GET_PARSE
04A3'CF DF 0387 1569 PUSHAL FLAGS ; TSTCNTRL flags
049B'CF DF 038B 1570 PUSHAL CONTROL_CASE ; main case flags
0910'CF 05 FB 038F 1571 CALLS #5,FILE_SBRTN ; see if we can start the process
0394 1572
0394 1573 ; and do next case
0394 1574
0040 31 0394 1575 BRW CONTIN_MAIN ; perform next case
0397 1576
0397 1577 ASSIGN_BLK:
0397 1578 ; reached this by GET_PARSE setting CASE$V_ASSIGN
0397 1579 ; handle the assignment
0397 1580
049B'CF DF 0397 1581 PUSHAL CONTROL_CASE ; main case flags
0EAO'CF 01 FB 039B 1582 CALLS #1,ASSIGN_SBRTN ; do the assignment
03A0 1583
03A0 1584 ; and do next case
03A0 1585
0034 31 03A0 1586 BRW CONTIN_MAIN ; perform next case
03A3 1587
03A3 1588 CREATE_PROC_BLK:
03A3 1589 ; reached this by FILE setting CASE$V_CREATE_PROC
03A3 1590 ; create the detached process
03A3 1591
0659'CF DF 03A3 1592 PUSHAL PARSE_FLAGS ; flags from parse routine
000C'CF DF 03A7 1593 PUSHAL TOTAL_PROC_LIMIT ; limit of allowed processes
4660'CF 7F 03AB 1594 PUSHAL CURRENT_LIST_HEAD ; head of currently running list
4650'CF 7F 03AF 1595 PUSHAL AVAIL_LIST_HEAD ; head of available slots
049F'CF DF 03B3 1596 PUSHAL CREATE_PROC_CASE ; case of type of proc to start
049B'CF DF 03B7 1597 PUSHAL CONTROL_CASE ; main case flags
04A3'CF DF 03BB 1598 PUSHAL FLAGS ; TSTCNTRL flags
0000'CF DF 03BF 1599 PUSHAL PROCESS_RUNNING ; number of processes running
0004'CF DF 03C3 1600 PUSHAL TOTAL_PROC_RUN ; number of processes ran so far
0962'CF 09 FB 03C7 1601 CALLS #9,CREATE_PROC_SBRTN ; create the process
03CC 1602
03CC 1603 ; and do next case
03CC 1604
0008 31 03CC 1605 BRW CONTIN_MAIN ; perform next case
```

```
03CF 1606
03CF 1607 GET_PARSE_BLK:
03CF 1608 ; reached by CASE$V_PARSE being set
03CF 1609 ; get a record and parse it
107D'CF 00 FB 03CF 1610
03D4 1611 CALLS #0,GET_PARSE_SBRIN ; parse the record
03D4 1612
03D4 1613 ; and do next case
0000 31 03D4 1614
03D7 1615 BRW CONTIN_MAIN ; perform next case
03D7 1616
03D7 1617 CONTIN_MAIN:
FF0E 31 03D7 1618 BRW MAIN_LOOP ; continue loop
03DA 1619
03DA 1620
03DA 1621
03DA 1622 END_MAIN:
03DA 1623 ; made it through, time to clean up
03DA 1624
03DA 1625 ; print ending time stamp
03DA 1626
04A3'CF 20 C8 03DA 1627 BISL2 #TCNTRL$M_WRT_MSG,FLAGS ; always write this message
03DF 1628
03DF 1629 PUSHL #0
0516'CF 00 DD 03DF 1629 PUSHAL PRC_NAM_DESC
02 DF 03E1 1630
00C31083 02 DD 03E5 1631
8F DD 03E7 1632
00000000'GF 04 FB 03ED 1633
03F4 1634
03F4 1635
03F4 1636 END_TCNTRL:
03F4 1637 ; do clean up
03F4 1638
03F4 1639
03F4 1640 $CANEXH_S ; cancel the exit handler
03FD 1641
03FD 1642 $SETAST_S ENBFLG = #0 ; disable any further AST's
0406 1643 $SETSFMS ENBFLG = #0 ; don't report sys service failure
040F 1644
040F 1645 $DASSGN_S CHAN = MBX_CHAN ; deassign the mailbox channel
041B 1646 $DASSGN_S CHAN = TTRCHAN ; deassign the terminal channel
0427 1647
0427 1648 $CLOSE FAB = DATA_FILE_FAB ; close data file
0432 1649 $CLOSE FAB = CNTRL_LOG_FAB ; close the perm log file
043D 1650
043D 1651 $SETPRN_S PRCNAM = PRCNAMDESC ; reset users process name
0448 1652
0448 1653 ; now, find out how we are leaving, and set exit accordingly
0448 1654
50 01 D0 0448 1655 MOVL #SS$NORMAL,R0 ; assume success
01 04A3'CF 0F E1 0448 1656
05 0451 1657 BBC #TCNTRL$V_EXIT_HAND,FLAGS,10$ ; did we come from exit handler?
0452 1658 RSB ; yes, so go back there
0452 1659
0452 1660 10$:
07 04A3'CF 0D E1 0452 1661 BBC #TCNTRL$V_ABORT,FLAGS,20$ ; is this an abort condition?
50 00C3002C 8F D0 0458 1662 MOVL #TCNTRL$ABORT,R0 ; yes, so signal that
```

TSTCNTRL
V04-000

TEST PACKAGE CONTROL PROGRAM
Main Program

H 4

16-SEP-1984 01:30:05
5-SEP-1984 04:26:03

VAX/VMS Macro V04-00
[UETP.SRC]UETPHAS00.MAR;1

Page 44
(29)

```

      045F 1663
      045F 1664 20$:
      045F 1665
50 10000000 8F C8 045F 1666 BLSL2 #STSSM_INHIB_MSG,R0 ; don't print message again
      0466 1667
04 0466 1668 RET ; time to go . . .
```

```
00000004 0467 1670      .SBTTL Abort TSTCNTRL Subroutine
00000008 0467 1671      ;
0467 1672      ; define offsets for parameters
0467 1673
0467 1674      ABT_FLAGS = 4
0467 1675      ABT_C_CASE = 8
0467 1676      ;++
0467 1677      : FUNCTIONAL DESCRIPTION:
0467 1678      :
0467 1679      : This subroutine is called when the TSTCNTRL must be abnormally
0467 1680      : terminated. The reasons for termination may be that a control-C has
0467 1681      : been typed by the user, an unexpected system service or RMS error has
0467 1682      : occurred, or the TSTCNTRL is being forced to exit by another process.
0467 1683      : This routine is reached by the CASE$V_ABORT bit being set. The routine
0467 1684      : terminates all currently running process, and clears all of the case
0467 1685      : flags (except for the PROC_TERM flag) and sets a flag (TCNTRL$V_ABORT)
0467 1686      : so that all other subroutines know that the TSTCNTRL is in an abort
0467 1687      : phase. Control then returns to the main case loop where it will stay
0467 1688      : until all of the currently running process have terminated.
0467 1689      :
0467 1690      : CALLING SEQUENCE:
0467 1691      :
0467 1692      :     PUSHAL CONTROL_CASE
0467 1693      :     PUSHAL  FLAGS
0467 1694      :     CALLS  #2,ABORT_TCNTRL_SBRTN
0467 1695      :
0467 1696      : INPUT PARAMETERS:
0467 1697      :
0467 1698      :     ABT_FLAGS(AP) - the address of the TSTCNTRL flags
0467 1699      :     ABT_C_CASE(AP) - the address of the main control case flags
0467 1700      :
0467 1701      : IMPLICIT INPUTS:
0467 1702      :
0467 1703      :     None.
0467 1704      :
0467 1705      : OUTPUT PARAMETERS:
0467 1706      :
0467 1707      :     None.
0467 1708      :
0467 1709      : IMPLICIT OUTPUTS:
0467 1710      :
0467 1711      :     The main control case flag will be cleared of everything except for
0467 1712      :     CASE$V_PROC_TERM.
0467 1713      :     The TCNTRL$V_ABORT flag will be set.
0467 1714      :
0467 1715      : COMPLETION CODES:
0467 1716      :
0467 1717      :     SSS_NORMAL
0467 1718      :
0467 1719      : SIDE EFFECTS:
0467 1720      :
0467 1721      :     This will cause the currently running processes to be aborted.
0467 1722      :     It will cause the case to only go to PROC_TERM_SBRTN.
0467 1723      : --
0467 1724
0004 0467 1725      ABORT_TCNTRL_SBRTN:
0004 0467 1726      .WORD ^M<R2>
```

				0469	1727	
				0469	1728	
				0469	1729	; clear the case flags of everything except the process termination
				0469	1730	; flag.
52	FFFFFFFF	8F	DO	0469	1731	MOVL #-1,R2 ; set all bits
	52	04	CA	0470	1732	BICL2 #CASE\$M_PROC_TERM,R2 ; clear the proc term bit
	08 BC	52	CA	0473	1733	BICL2 R2,@ABT_C_CASE(AP) ; and clear everything but
				0477	1734	; process term bit
				0477	1735	
				0477	1736	; clear the process pending bit so we don't start it
				0477	1737	
04 BC	00001000	8F	CA	0477	1738	BICL2 #TCNTRL\$M_PROC_PEND,@ABT_FLAGS(AP) ; no processes pending to start
				047F	1739	
				047F	1740	; print out a message saying that we are going to abort
				047F	1741	
	00F0'CF	7F	047F	1742	PUSHAQ ABORT_MSG ; say we aborting	
	01	DD	0483	1743	PUSHL #1 ;	
	00C31130	8F	DD	0485	1744	PUSHL #TCNTRL\$TEXT!STSSK_WARNING ; print it as a warning
00000000'GF	03	FB	048B	1745	CALLS #3,G^LIB\$SIGNAL ; write the message	
				0492	1746	
				0492	1747	; now stop all currently running processes
				0492	1748	
	13DB'CF	00	FB	0492	1749	CALLS #0,ABORT_PROCESSES ; abort all detached processes
				0497	1750	
				0497	1751	; we are now in abort phase, so let everyone else know
				0497	1752	
04 BC	00000800	8F	C8	0497	1753	BISL2 #TCNTRL\$M_EXIT,@ABT_FLAGS(AP) ; exit when we can
				049F	1754	
	50	01	DO	049F	1755	MOVL #SS\$_NORMAL,R0 ; return success
				04A2	1756	
			04	04A2	1757	RET
				04A3	1758	

```
00000004 04A3 1760 .SBTTL Timer Expiration Subroutine
00000008 04A3 1761
0000000C 04A3 1762 ; define offsets for parameters
04A3 1763
04A3 1764 TIM_FLAGS = 4
04A3 1765 TIM_C CASE = 8
04A3 1766 TIM_PRSE = 12
04A3 1767
04A3 1768 ;++
04A3 1769 : FUNCTIONAL DESCRIPTION:
04A3 1770 :
04A3 1771 : This subroutine is reached when the timer (MAXTIME) to prevent processes
04A3 1772 : from hanging the TSTCNTRL expires. The expiration of the timer sets a
04A3 1773 : case flag, CASE$V_TIME_EXP, via the timer AST routine (TIME_EXP_AST).
04A3 1774 : This routine then writes a message, aborts all currently running
04A3 1775 : processes, and sets a TSTCNTRL flag (TCNTRL$V_TIME_EXP) to signal other
04A3 1776 : routines that the timer has expired.
04A3 1777 :
04A3 1778 : CALLING SEQUENCE:
04A3 1779 :
04A3 1780 : PUSHAL PARSE_FLAGS
04A3 1781 : PUSHAL CONTROL_CASE
04A3 1782 : PUSHAL FLAGS
04A3 1783 : CALLS #2, TIME_EXP_SBRTN
04A3 1784 :
04A3 1785 : INPUT PARAMETERS:
04A3 1786 :
04A3 1787 : TIM_FLAGS(AP) - the address of the TSTCNTRL flags
04A3 1788 : TIM_C CASE(AP) - the address of the main control case flags
04A3 1789 : TIM_PRSE(AP) - the address of the parse flags
04A3 1790 :
04A3 1791 : IMPLICIT INPUTS:
04A3 1792 :
04A3 1793 : None.
04A3 1794 :
04A3 1795 : OUTPUT PARAMETERS:
04A3 1796 :
04A3 1797 : None.
04A3 1798 :
04A3 1799 : IMPLICIT OUTPUTS:
04A3 1800 :
04A3 1801 : The flag TCNTRL$V_TIME_EXP will be set.
04A3 1802 : The flag TCNTRL$V_ABLE_TO_WRAP is cleared.
04A3 1803 :
04A3 1804 : COMPLETION CODES:
04A3 1805 :
04A3 1806 : SSS_NORMAL
04A3 1807 :
04A3 1808 : SIDE EFFECTS:
04A3 1809 :
04A3 1810 : The TCNTRL$V_TIME_EXP bit being set will cause the GET_PARSE_SBRTN to
04A3 1811 : keep getting records until the end of the current segment. A segment
04A3 1812 : is defined to be ended when either the next process to be started is
04A3 1813 : a "runs alone" process (i.e., when PARSE$V_RUNS_OTHERS = 0), or when
04A3 1814 : the end-of-file is reached.
04A3 1815 :
04A3 1816 :
```

```
0000 04A3 1817 :-- Wrapping of the data file is disabled.
04A3 1818 :--
04A3 1819
04A3 1820 TIME_EXP_SBRTN:
04A3 1821 .WORD ^M<>
04A5 1822
04A5 1823 ; clear out the parse flag which got us here, and turn off wrapping
04A5 1824
04A5 1825 BICL2 #CASE$M_TIME_EXP,@TIM_C_CASE(AP) ; don't case back here
08 BC 08 BC 02 CA 04A9 1826 BISL2 #CASE$M_PARSE,@TIM_C_CASE(AP) ; but do parse instead
00000080 8F C8 04B1 1827
04 BC 10 CA 04B1 1828 BICL2 #TCNTRL$M_ABLE_TO_WRAP,@TIM_FLAGS(AP) ; wrapping no longer allowed
04B5 1829
04B5 1830 ; now, print out message saying that the timer has expired
04B5 1831
02A4'CF 7F 04B5 1832 PUSHAQ TIME_OUT_ERR ; time was exceeded
01 DD 04B9 1833 PUSHL #1
00C31130 8F DD 04BB 1834 PUSHL #TCNTRL$TEXT!STSSK_WARNING
00000000'GF 03 FB 04C1 1835 CALLS #3,G^LIB$SIGNAL ; write the message
04C8 1836
04C8 1837 ; set bit saying that timer expired
04C8 1838
04 BC 00004000 8F C8 04C8 1839 BISL2 #TCNTRL$M_TIME_EXP,@TIM_FLAGS(AP) ; say that timer expired
04D0 1840
04D0 1841 ; now abort all currently running processes
04D0 1842
13DB'CF 00 FB 04D0 1843 CALLS #0,ABORT_PROCESSES ; abort all detached processes
04D5 1844
04D5 1845 ; now, check to see if the record we just parsed is an End-of-segment.
04D5 1846 ; That is, a process to be run by itself, or an EOF. If it is EOS, then
04D5 1847 ; goto the proper routine, else goto GET_PARSE.
04D5 1848
0A 04 BC 0B E1 04D5 1849 BBC #TCNTRL$V_EXIT,@TIM_FLAGS(AP),10$ ; not time to exit, so see if
04DA 1850 ; process is pending
04DA 1851
04DA 1852 ; we have an end-of-file, so turn off parse
04DA 1853
08 BC 00000080 8F CA 04DA 1854 BICL2 #CASE$M_PARSE,@TIM_C_CASE(AP) ; EOF, so turn off parse
16 11 04E2 1855 BRB 20$ ; and leave turning off time expo.
04E4 1856
04E4 1857 10$:
04E4 1858
04E4 1859 ; do we have a process pending to be run?
19 04 BC 0C E1 04E4 1860 BBC #TCNTRL$V_PROC_PEND,@TIM_FLAGS(AP),30$ ; no proc pending, exit
04E9 1861
04E9 1862 ; we have a process pending, is it to be run alone?
04E9 1863
14 0C BC 00 E0 04E9 1864 BBS #PARSE$V_PROC_RUNS_OTHERS,@TIM_PRSE(AP),30$ ; proc not seq, exit
04EE 1865
04EE 1866 ; we have a process to be run alone, prepare it to run
04EE 1867
08 BC 10 C8 04EE 1868 BISL2 #CASE$M_FILE,@TIM_C_CASE(AP) ; try and start process now
04F2 1869
04 BC 00001000 8F CA 04F2 1870 BICL2 #TCNTRL$M_PROC_PEND,@TIM_FLAGS(AP) ; clear pending flag
04FA 1871
04FA 1872 20$:
04FA 1873 ; End-of-segment, so turn off time expired flag
```

TSTCNTRL
V04-000

TEST PACKAGE CONTROL PROGRAM
Timer Expiration Subroutine

M 4

16-SEP-1984 01:30:05 VAX/VMS Macro V04-00
5-SEP-1984 04:26:03 [JETP.SRC]UETPHAS00.MAR;1

Page 49
(31)

04 BC	00004000	8F	CA	04FA	1874		
				04FA	1875		
	50	01	D0	0502	1876	30\$:	BICL2
				0502	1877		MOVL
				0505	1878		
			04	0505	1879		RET

#TCNTRL\$M_TIME_EXP,@TIM_FLAGS(AP) ; since EOS, clear expired flag
#SS\$ _NORMAL,R0 ; return success

```
0506 1881
0506 1882      .SBTTL Process Termination Subroutine
0506 1883
0506 1884      ; define offsets for parameters
0506 1885
00000004 0506 1886      P_CMPLT_Q_HEAD = 4
00000008 0506 1887      P_PROCESS_RUN = 8
0000000C 0506 1888      P_TERM_FLG = 12
00000010 0506 1889      P_CNTRL_CASE = 16
00000014 0506 1890      STRT_DESC = 20
00000018 0506 1891      STP_DESC = 24
0000001C 0506 1892      P_AVL_LST_HEAD = 28
0506 1893
0506 1894      ;++
0506 1895      : FUNCTIONAL DESCRIPTION:
0506 1896      :
0506 1897      : This subroutine is utilized whenever a detached process is completed.
0506 1898      : It determines whether or not the detached process succeeded and
0506 1899      : it will print to the console and the log file the proper records from
0506 1900      : the process' temporary log file.
0506 1901
0506 1902      : CALLING SEQUENCE:
0506 1903      :
0506 1904      : This is called when CASE$V_PROC_TERM is set (set by a previous AST).
0506 1905      : PUSHAQ AVAIL_LIST_HEAD
0506 1906      : PUSHAQ STOP_DESC
0506 1907      : PUSHAQ START_DESC
0506 1908      : PUSHAL CNTRL_CASE
0506 1909      : PUSHAL FLAGS
0506 1910      : PUSHAL PROCESS_RUNNING
0506 1911      : PUSHAQ COMPLETE_Q_HEAD
0506 1912      : CALLS #7,PROCESS_TERM_SBRTN
0506 1913
0506 1914      : INPUT PARAMETERS:
0506 1915      :
0506 1916      : P_CMPLT_Q_HEAD(AP) - address of quadword header of the completed process
0506 1917      : queue.
0506 1918      : P_PROCESS_RUN(AP) - address of a longword holding a count of the currently
0506 1919      : running processes (including this one).
0506 1920      : P_CNTRL_CASE(AP) - address of a longword holding the CNTRL_CASE flags.
0506 1921      : P_TERM_FLG(AP) - address of a longword holding the TSTCNTRL flags.
0506 1922      : STRT_DESC(AP) - address of a quadword descriptor of the beginning sentinel.
0506 1923      : STP_DESC(AP) - address of a quadword descriptor of the ending sentinel.
0506 1924      : P_AVL_LST_HEAD(AP) - address of a quadword header of the available list.
0506 1925
0506 1926      : IMPLICIT INPUTS:
0506 1927      :
0506 1928      : Termination information off of the completed process queue.
0506 1929
0506 1930      : OUTPUT PARAMETERS:
0506 1931      :
0506 1932      : None
0506 1933
0506 1934      : IMPLICIT OUTPUTS:
0506 1935      :
0506 1936      : None
0506 1937
```

```
0506 1938 : COMPLETION CODES:
0506 1939 :
0506 1940 :     SSS_NORMAL
0506 1941 :
0506 1942 : SIDE EFFECTS:
0506 1943 :
0506 1944 :     May change the flags PROCINFO$V_MBX_CHAN, CASE$V_PROCTERM,
0506 1945 :     CASE$V_FILE, and TCNTRL$V_PROC_PEND.
0506 1946 : --
0506 1947 :
0506 1948 :
0506 1949 : PROCESS_TERM_SBRN:
OFFC 0506 1950 : .WORD ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
0506 1951 :
0508 1952 : ; assume we don't have to come back here.
0508 1953 :
10 BC 04 CA 0508 1954 : BICL2 #CASE$M_PROC_TERM,@P_CNTRL_CASE(AP) ; clear flag to here
050C 1955 :
050C 1956 : ; assume we want to parse again
10 BC 00000080 8F C8 050C 1957 :
050C 1958 : BISL2 #CASE$M_PARSE,@P_CNTRL_CASE(AP) ; assume we parse again
0514 1959 :
0514 1960 : ; however, we don't want to parse if either EXIT v CREATE_PROC v
0514 1961 : ; ABORT are set.
0514 1962 :
0C 0C BC 0B E0 0514 1963 : BBS #TCNTRL$V_EXIT,@P_TERM_FLG(AP),3$ ; if exit, then clear parse
0519 1964 :
07 10 BC 06 E0 0519 1965 : BBS #CASE$V_CREATE_PROC,@P_CNTRL_CASE(AP),3$ ; if create, clear parse
051E 1966 :
02 0C BC 0D E0 051E 1967 : BBS #TCNTRL$V_ABORT,@P_TERM_FLG(AP),3$ ; if abort, clear parse
0523 1968 :
0523 1969 : BRB 5$ ; none were set, parse next
0525 1970 :
10 BC 00000080 8F CA 0525 1971 3$:
0525 1972 : BICL2 #CASE$M_PARSE,@P_CNTRL_CASE(AP) ; don't parse again
052D 1973 :
052D 1974 5$:
052D 1975 : ; initialize the stack area for local variables
052D 1976 :
SE 00000259 8F C2 052D 1977 :
0534 1978 :
6E 0259 8F 00 00 8F 00 2C 0534 1979 :
5A 5E D0 053D 1980 :
0540 1981 :
0540 1982 : ; see if any more completed processes are queued up
0540 1983 :
0540 1984 : ; turn off AST's to avoid a race condition
0540 1985 :
0540 1986 : $SETAST_S ENBFLG = #0 ; disable AST's
0549 1987 : CLRL R7 ; assume it was disabled
09 50 D1 054B 1988 : CMPL R0,#SS$_WASSET ; were AST's enabled?
03 12 054E 1989 : BNEQ 10$ ; no, so continue
57 01 D0 0550 1990 : MOVL #1,R7 ; yes, so save it to be reenabled
0553 1991 :
0553 1992 10$:
0553 1993 : ; now, remove entry from head of queue and get address of entry
0553 1994 :
```

```
5B 04 BC 5E 0553 1995 REMQHI @P_CMPLT_Q_HEAD(AP),R11 ; this is the address of entry
                                0557 1996
                                0557 1997 ; if queue is empty then turn off CASE$V_PROC_TERM flag, otherwise
                                0557 1998 ; make sure it is set.
                                0557 1999
                                04 13 0557 2000 BEQL 20$ ; we just removed the last entry
                                0559 2001
10 BC 04 C8 0559 2002 BISL2 #CASE$M_PROC_TERM,@P_CNTRL_CASE(AP) ; there are still entries on
                                055D 2003 ; the queue, so come back to this
                                055D 2004 ; routine when done.
                                055D 2005
                                055D 2006 20$:
                                055D 2007 ; now reset AST's
                                055D 2008
                                055D 2009 $SETAST_S ENBFLG = R7 ; set AST to previous state
                                0566 2010
                                0566 2011
                                0566 2012 ; change the count of processes running
                                0566 2013
                                08 BC D7 0566 2014 DECL @P_PROCESS_RUN(AP) ; one less process running
                                0569 2015
                                0569 2016 ; now see if timer should be canceled
                                0569 2017
                                13 12 0569 2018 BNEQ 30$ ; not zero, so just continue
                                056B 2019 $CANTIM_S REQIDT = #MAXTIME_ID ; none running, so cancel timer
0C BC 00000100 8F CA 0576 2020 BICL2 #TCNTRL$M_TIMER_SET,@P_TERM_FLG(AP) ; no timer running, so clear fla
                                057E 2021
                                057E 2022 30$:
                                057E 2023 ; see if a mailbox was made to pass parameters to an image, if one
                                057E 2024 ; was made, deassign the channel
                                057E 2025
                                08 00 057E 2026 BBCC #PRCINFO$V_MBX_CHAN,- ; is there a channel?
0B 38 AB E5 0580 2027 PRCINFO$L_FLAGS(R11),40$
                                0583 2028
                                0583 2029 $DASSGN_S CHAN = PRCINFO$W_MBX_CHAN(R11) ; yes, deassign it
                                058E 2030
                                058E 2031 40$:
                                058E 2032 ; make the termination file descriptor
                                058E 2033
                                60 AB 3C 058E 2034 MOVZWL PRCINFO$W_FILNAM_SIZ(R11),-
0225 CA 0591 2035 TRMNTN_FIL_DESC(R10) ; get the size of filename
                                0594 2036
                                4C AB D0 0594 2037 MOVL PRCINFO$A_FILNAM(R11),-
0229 CA 0597 2038 <DSC$A_POINTER+TRMNTN_FIL_DESC>(R10) ; point to filename
                                059A 2039
                                059A 2040 ; initialize the descriptor area for the temporary command file
                                059A 2041
                                08 AA DE 059A 2042 MOVAL TRMNTN_COM_STR(R10),- ; put address of string,
04 AA 059D 2043 <TRMNTN_COM_DESC+DSC$A_POINTER>(R10) ; in pointer area
                                059F 2044
                                059F 2045 ; make the filename for the temporary command file
                                059F 2046
                                0225 CA /F 059F 2047 PUSHAQ TRMNTN_FIL_DESC(R10) ; the test filename
                                3E AB 3F 05A3 2048 PUSHAQ PRCINFO$W_ID_NUMB(R11) ; the ID number
                                0467 CF 7F 05A6 2049 PUSHAQ TMP_COM_EXT_DESC ; the file extension
                                6A 7F 05AA 2050 PUSHAQ TRMNTN_COM_DESC(R10) ; where the new filename goes
0E35 CF 04 FB 05AC 2051 CALLS #4,MAKE_TMP_LOG_NAM ; and make the name
```

```
05B1 2052
05B1 2053 ; delete the temporary command log file (if any), use TMP_LOG_RAB
05B1 2054
05B1 2055 $FAB_STORE FAB = DEL_COM_FAB,-
05B1 2056 FNA = TRMNTN_COM_STR(R10),- ; the file name is already made
05B1 2057 FNS = TRMNTN_COM_DESC(R10)
05BF 2058
05BF 2059 $ERASE FAB = DEL_COM_FAB,-
05BF 2060 ERR = RMSERR ; erase the file,
05CE 2061
05CE 2062 ; initialize descriptor area for temporary log file name
05CE 2063
0235 CA DE 05CE 2064 MOVAL TRMNTN_LOG_STR(R10),- ; put the address in the
0231 CA 05D2 2065 <TRMNTN_LOG_DESC+DSC$A_POINTER>(R10) ; pointer area
05D5 2066
05D5 2067 ; now make the temporary log file name
05D5 2068
0225 CA 7F 05D5 2069 PUSHAQ TRMNTN_FIL_DESC(R10) ; the test file name
3E AB 3F 05D9 2070 PUSHAQ PRCINFO$W_ID_NUMB(R11) ; the ID number
041C CF 7F 05DC 2071 PUSHAQ LOG_EXT_DESC ; the temp log extension
022D CA 7F 05E0 2072 PUSHAQ TRMNTN_LOG_DESC(R10) ; the temp log file
OE35 CF 04 FB 05E4 2073 CALLS #4,MAKE_TMP_LOG_NAM ; and make the log name
05E9 2074
05E9 2075 ; now adjust FAB to hold this log filename
05E9 2076
05E9 2077 $FAB_STORE FAB = TMP_LOG_FAB,- ; make FAB have log file name
05E9 2078 FNS = TRMNTN_LOG_DESC(R10),-
05E9 2079 FNA = TRMNTN_LOG_STR(R10)
05FA 2080
05FA 2081 ; initialize the descriptor area for the return process name
05FA 2082
024A CA DE 05FA 2083 MOVAL RTN_NAM_STR(R10),- ; put the address in the
0246 CA 05FE 2084 <RTN_NAM_DESC+DSC$A_POINTER>(R10) ; pointer area
0601 2085
0601 2086 ; now make the return process name
0601 2087
0225 CA DF 0601 2088 PUSHAQ TRMNTN_FIL_DESC(R10) ; the test file name
3E AB DF 0605 2089 PUSHAQ PRCINFO$W_ID_NUMB(R11) ; the ID number
0242 CA DF 0608 2090 PUSHAQ RTN_NAM_DESC(R10) ; the return process name
ODC7 CF 03 FB 060C 2091 CALLS #3,MAKE_PROC_NAM ; and make the process name
0611 2092
0611 2093 CHK_ABT_PROC:
0611 2094 ; see if the process was aborted by the TSTCNTRL. If it was, then
0611 2095 ; check the timer which watches to see if a process aborts and
0611 2096 ; then print out an abort message and skip the rest
0611 2097
03 01 E0 0611 2098 BBS #PRCINFO$V_PROC_ABORTED,- ; aborted, so report error
03 38 AB 0613 2099 PRCINFO$L_FLAGSTR11),5$
0062 31 0616 2100
0619 2101 BRW CHK_ERROR ; not abort, continue
0619 2102
0619 2103 5$:
0619 2104 ; process was aborted, so check time and print out a message
0619 2105
0619 2106 $CANTIM_S REQIDT = #ABRT_TIM_ID ; cancel the current timer
0624 2107
0624 2108 ; w, see if we are to reset the timer
```

```

      0624 2109
      10 E0 0624 2110      BBS      #TCNTRL$V PROC_NUKED,-      ; if $DELPRC has been used
      0C BC 0626 2111      @P_TERM_FLG(AP),-      ; to nuke the processes, then
      22 0628 2112      20$      ; don't reset the timer
      0B BC 0629 2113
      0A D5 0629 2114      TSTL      @P_PROCESS_RUN(AP)      ; are we at 0 running processes?
      12 062C 2115      BNEQ      10$      ; no, so reset the timer
      00010000 8F CA 062E 2116
      0C BC 062E 2117      BICL2     #TCNTRL$M PROC_NUKED,-      ; yes, so clear out process
      13 11 0634 2118      @P_TERM_FLG(AP)      ; nuked flag and
      0636 2119      BRB      20$      ; don't reset timer
      0638 2120
      0638 2121 10$:
      0638 2122      $SETIMR_S DAYTIM = PROC_TERM_DELT,-      ; reset timer to insure that at
      0638 2123      ASTADR = NUKE_PROC_AST,-      ; least one process terminates
      0638 2124      REQIDT = #ABRT_TIM_ID      ; in a reasonable time
      064B 2125
      0C BC 20 CA 064B 2126 20$:
      064F 2127      BICL2     #TCNTRL$M_WRT_MSG,@P_TERM_FLG(AP) ; don't print if short report
      013C'CF 7F 064F 2128
      01 DD 0653 2129      PUSHAQ    TCNTRL_ABORT_MSG      ; say TSTCNTRL aborted this
      00C31130 8F DD 0655 2131      PUSHL     #1
      10 009E'CF FO 065B 2132      PUSHL     #TCNTRL$ TEXT!STSS$K_WARNING
      0C 0660 2133      INSV      FAC_CODE,#STSS$V_FAC_NO,-      ; set the facility code of the
      6E 0661 2134      (SP)      ; message to be the defined
      10 AB 7F 0662 2135      PUSHAQ    PRCINFO$Q TERMTIME(R11) ; facility code
      0242 CA 7F 0665 2136      PUSHAQ    RTN_NAM_DESC(R10)      ; time it was aborted
      02 DD 0669 2137      PUSHL     #2      ; name of the process
      00C310E0 8F DD 066B 2138      PUSHL     #TCNTRL$ ABENDD!STSS$K_WARNING ; give an abort warning
      00000000'GF 07 FB 0671 2139      CALLS     #7,G^LIB$SIGNAL      ; write message
      01DD 31 0678 2140
      0678 2141      BRW      RETURN_BLK      ; and leave
      067B 2142
      067B 2143      CHK_ERROR:
      067B 2144      ; find out if the process returned successfully or not. If it did,
      067B 2145      ; then look for a log file; if it didn't, report the problem first.
      067B 2146
      10000000 8F CB 067B 2147      BICL3     #STSS$M_INHIB_MSG,-      ; set the status to print
      53 08 AB 0681 2148      PRCINFO$L_FINALSTS(R11),R3
      5D 53 EB 0684 2149      BLBS      R3,CHK_LOG_FIL      ; BR if there was no error
      0687 2150
      0687 2151      ; it did not complete successfully, determine the problem and
      0687 2152      ; print it out
      0687 2153
      52 0242 CA 7E 0687 2154      MOVAQ     RTN_NAM_DESC(R10),R2      ; get address of process name
      068C 2155
      068C 2156      $FAO_S     CTRSTR = BAD_STATUS,-
      068C 2157      OUTLEN = FAO_BUF,-
      068C 2158      OUTBUF = BUFFER_PTR,-
      068C 2159      P1 = R2
      06A1 2160
      06A1 2161      $GETMSG_S MSGID = R3,-      ; get message that goes with
      06A1 2162      MSGLEN = GETMSG_PTR,-      ; this status
      06A1 2163      BUFADR = GETMSG_DESC,-
      06A1 2164      FLAGS = #15,-
      06A1 2165      OUTADR = MSG_BLOCK
```

```
05C9'CF DF 06B8 2166 PUSHAL GETMSG_PTR ; send descriptor address ...
08 DD 06BC 2167 PUSHL #8
00010002 8F DD 06BE 2168 PUSHL #^X10002 ; and the error message
007480B8 8F DD 06C4 2169 PUSHL #UETP$ COPY_LOG_LINE-STSSK_SUCCESS+STSSK_WARNING
10 009E'CF FO 06CA 2170 INSV FAC_CODE,#STSSV-FAC_NO,- ; set the facility code of the
6E 0C 06CF 2171 #STSS$ FAC_NO,(SP) ; message to be the defined
052D'CF DF 06D1 2172 PUSHAL FAO_BUF
01 DD 06D5 2173 PUSHL #1
00C31130 8F DD 06D7 2174 PUSHL #TCNTRL$ TEXT!STSSK_WARNING
0000C700'GF 07 FB 06DD 2175 CALLS #7,G^LIB$SIGNAL
06E4 2176
06E4 2177 CHK_LOG_FIL:
06E4 2178 ; now, check to see if the log file is there. If it is, then look
06E4 2179 ; at the records in it. If it isn't, then leave.
06E4 2180
06E4 2181 $OPEN FAB = TMP_LOG_FAB,- ; open the temp log file
06E4 2182 ERR = RMSERR
06F3 2183
50 00000000'8F D1 06F3 2184 CMPL #RMS$ FNF,R0 ; is there a log file
03 12 06FA 2185 BNEQ PROCESS_CMPLT ; yes, so look at it
06FC 2186
013F 31 06FC 2187 BRW CLEAN_OUT ; no log file, so leave
06FF 2188
06FF 2189
06FF 2190 PROCESS_CMPLT:
06FF 2191 ; the process completed with a log file (may have completed successfully
06FF 2192 ; or with error), print out the contents of the logfile
06FF 2193
06FF 2194 ; connect the data stream to the temporary log file,
06FF 2195 ; and pull records out of it
06FF 2196
06FF 2197 $CONNECT RAB = TMP_LOG_RAB,- ; connect stream to temp log
06FF 2198 ERR = RMSERR ; file
070E 2199
070E 2200 ; make the termination filename uppercase, so that we can use it as
070E 2201 ; part of the sentinels.
070E 2202
0225 CA 7F 070E 2203 PUSHAQ TRMNTN_FIL_DESC(R10)
0225 CA 7F 0712 2204 PUSHAQ TRMNTN_FIL_DESC(R10)
00000000'GF 02 FB 0716 2205 CALLS #2,G^STR$UPCASE ; change filename to upper case
071D 2206
071D 2207 ; make descriptor for the log file records
071D 2208
1D AA DE 071D 2209 MOVAL LOG_RCRD_STR(R10),- ; put address in the
19 AA 0720 2210 <LOG_RCRD_DESC+DS($A_POINTER)>(R10) ; pointer area
0722 2211
0722 2212 ; make descriptor for temporary buffer for log file records
0722 2213
0125 CA DE 0722 2214 MOVAL TEMP_BUFF_STR(R10),- ; put address in the
0121 CA 0726 2215 <TEMP_BUFF_DESC+DS($A_POINTER)>(R10) ; pointer area
0729 2216
0729 2217 GET_RCRD:
0729 2218 ; now start getting records out of the temporary log file
0729 2219
0729 2220
0729 2221 ; set up TMP_LOG_RAB to put the newly fetched record in the right place
0729 2222
```

			0729	2223	\$RAB_STORE RAB = TMP_LOG_RAB,-	; initialize the temp. log file RAB
			0729	2224	UBF = LOG_RCRD_STR(R10)	; put the record here
			0733	2225		
			0733	2226	\$GET RAB = TMP_LOG_RAB,-	
			0733	2227	ERR = RMSERR	
			0742	2228		
00000000'8F	50	D1	0742	2229	CMPL R0, #RMS\$ EOF	; was there an end-of-file?
	03	12	0749	2230	BNEQ 10\$	; no, so continue
	00CD	31	074B	2231	BRW TRMNTN_END	; yes, so exit
			074E	2232		
			074E	2233	10\$:	
			074E	2234		; we have a record, so finish making the descriptor for it and
			074E	2235		; convert it to uppercase
			074E	2236		
0B96'CF		3C	074E	2237	MOVZWL <TMP_LOG_RAB+RAB\$W_RSZ>,-	
15 AA			0752	2238	LOG_RCRD_DESC(R10)	; put size in descriptor
			0754	2239		
15 AA	7F		0754	2240	PUSHAQ LOG_RCRD_DESC(R10)	
15 AA	7F		0757	2241	PUSHAQ LOG_RCRD_DESC(R10)	
00000000'GF	02	FB	075A	2242	CALLS #2,G^STR\$UPCASE	; change record to upper case
			0761	2243		
			0761	2244		; look for the beginning sentinel so that we may start copying records
			0761	2245		
57	14 AC	D0	0761	2246	MOVL STRT_DESC(AP),R7	; temporarily keep the address of
			0765	2247		; the beginning sentinel
			0765	2248		
04 B7	67	39	0765	2249	MATCHC DSC\$W_LENGTH(R7),@DSC\$A_POINTER(R7),-	
15 AA			0769	2250	LOG_RCRD_DESC(R10),-	
1D AA			076B	2251	LOG_RCRD_STR(R10)	; is there a begin sentinel?
			076D	2252		
	BA	12	076D	2253	BNEQ GET_RCRD	; no, so keep looking for one
			076F	2254		
			076F	2255		; we found the beginning sentinel, now see if the filename is there
			076F	2256		; (to make sure that this record is indeed a sentinel record and not
			076F	2257		; simply a data record).
			076F	2258		
51	0225 CA	7E	076F	2259	MOVAQ TRMNTN FIL_DESC(R10),R1	; get address of term file
04 B1	61	39	0774	2260	MATCHC DSC\$W_LENGTH(R1),@DSC\$A_POINTER(R1),-	
15 AA			0778	2261	LOG_RCRD_DESC(R10),-	
1D AA			077A	2262	LOG_RCRD_STR(R10)	; is filename in same record?
	AB	12	077C	2263	BNEQ GET_RCRD	; no, so start over again
			077E	2264		
			077E	2265	COPY_RCRD:	
			077E	2266		; we found a begin sentinel, now copy data records to SYS\$OUTPUT
			077E	2267		; and perm log file
			077E	2268		
			077E	2269	\$GET RAB = TMP_LOG_RAB,-	; get next record
			077E	2270	ERR = RMSERR	
			078D	2271		
50	00000000'8F	D1	078D	2272	CMPL #RMS\$ EOF,R0	; was error end-of-file?
	03	12	0794	2273	BNEQ 10\$	; no, so continue
			0796	2274		
	0082	31	0796	2275	BRW TRMNTN_END	; EOF, so clean up and leave
			0799	2276		
			0799	2277	10\$:	
			0799	2278		; we have a record, so finish making the descriptor for it and
			0799	2279		; convert it to uppercase

OB96'CF	3C	0799	2280	
15 AA		0799	2281	MOVZWL <TMP_LOG_RAB+RAB\$W_RSZ>,-
		079D	2282	LOG_RCRD_DESC(R10) ; put size in descriptor
		079F	2283	
		079F	2284	; since this may be a valid data record, we don't want to change
		079F	2285	; the output version of the record to uppercase. We will put
		079F	2286	; the uppercase record in a temporary buffer to check the sentinel,
		079F	2287	; and output the original record.
		079F	2288	
15 AA	DO	079F	2289	MOVL LOG_RCRD_DESC(R10),-
011D CA		07A2	2290	TEMP_BUFF_DESC(R10) ; put the size of buffer in
		07A5	2291	; descriptor for STR\$UPCASE
15 AA	7F	07A5	2292	PUSHAQ LOG_RCRD_DESC(R10) ; this is the original record
011D CA	7F	07A8	2293	PUSHAQ TEMP_BUFF_DESC(R10) ; and this is the temporary one
00000000'GF	02	FB	07AC	CALLS #2,G*STR\$OPCASE ; change record to upper case
		07B3	2295	
		07B3	2296	
		07B3	2297	
		07B3	2298	; now see if we are at the end by looking for and ending sentinel
57 18 AC	DO	07B3	2299	MOVL STP_DESC(AP),R7 ; temporarily hold address of the
		07B7	2300	; ending sentinel
		07B7	2301	
04 B7 67	39	07B7	2302	MATCHC DSC\$W_LENGTH(R7),@DSC\$A_POINTER(R7),-
011D CA		07BB	2303	TEMP_BUFF_DESC(R10),-
0125 CA		07BE	2304	TEMP_BUFF_STR(R10) ; is there an ending sentinel?
		07C1	2305	
11 12		07C1	2306	BNEQ 20\$; no, so copy this record
		07C3	2307	
		07C3	2308	; there was an ending sentinel, so check for the filename to insure
		07C3	2309	; that this is a sentinel record
		07C3	2310	
51 0225 CA	DE	07C3	2311	MOVAL TRMNTN_FIL_DESC(R10),R1 ; get address of term file
		07C8	2312	
04 B1 61	39	07C8	2313	MATCHC DSC\$W_LENGTH(R1),@DSC\$A_POINTER(R1),-
011D CA		07CC	2314	TEMP_BUFF_DESC(R10),-
0125 CA		07CF	2315	TEMP_BUFF_STR(R10) ; is filename in same record?
		07D2	2316	
47 13		07D2	2317	BEQL TRMNTN_END ; yes, so exit
		07D4	2318	
		07D4	2319	
		07D4	2320	
		07D4	2321	20\$: ; we have a valid data record, so print it to the proper places
		07D4	2322	
009F	30	07D4	2323	BSBW INDENT_LOG_RECORD ; indent it from left margin
		07D7	2324	
		07D7	2325	; first write it to the console (SYS\$OUTPUT)
		07D7	2326	
15 AA	DF	07D7	2327	PUSHAL LOG_RCRD_DESC(R10) ; this is the record
00010001 8F	DD	07DA	2328	PUSHL #^XT0001
00C31131 8F	DD	07E0	2329	PUSHL #ICNTRL\$TEXT!ST\$K_SUCCESS
	DD	07E6	2330	PUSHL #3
50 5E	DO	07E8	2331	MOVL SP,R0
		07EB	2332	\$PUTMSG,S MSGVEC = (R0) ; write to SYS\$OUTPUT
		07FA	2333	
		07FA	2334	; and then write it to the permanent log file
		07FA	2335	
		07FA	2336	\$RAB_STORE RAB = CNTRL_LOG_RAB,-

```
07FA 2337          RSZ = LOG_RCRD_DESC(R10),-      ; size to be output
07FA 2338          RBF = LOG_RCRD_STR(R10)          ; where located
0809 2339
0809 2340          $PUT      RAB = CNTRL_LOG_RAB,-      ; write to perm file
0809 2341          ERR = RMSERR
0818 2342
FF63 31 0818 2343          BRW      COPY_RCRD          ; get next record
0818 2344
0818 2345          TRMNTN_END:
0818 2346          ; all through with the temporary log file, so clean up
0818 2347
0818 2348          $CLOSE      FAB = TMP_LOG_FAB,-          ; close the temporary log file
0818 2349          ERR = RMSERR
0818 2350
082A 2351          ; now see if we are to delete the temporary log file
082A 2352
082A 2353          BBC      #TCNTRL$V_DELETE_TEMP_LOG,-
OF 0C BC 02 E1 082C 2355          @P_TERM_FLG(AP),CLEAN_OUT      ; don't delete it, so continue
082F 2356
082F 2357          ; we are to delete it, so do it
082F 2358
082F 2359          $ERASE      FAB = TMP_LOG_FAB,-          ; delete the temp. log file
082F 2360          ERR = RMSERR
083E 2361
083E 2362          CLEAN_OUT:
083E 2363          ; all done, so print out ending message
083E 2364
0C BC 20 CA 083E 2366          BICL2      #TCNTRL$M_WRT_MSG,@P_TERM_FLG(AP) ; don't write this to the
0842 2367          ; console if it is short report
0842 2368
0842 2369          PUSHAL      PRCINFO$Q_TERMTIME(R11)      ; addr of time process ended
0242 CA 7F 0845 2370          PUSHAQ      RTN_NAM_DESC(R10)      ; the process name
02 DD 0849 2371          PUSHL      #2
00C31083 8F DD 084B 2372          PUSHL      #TCNTRL$ ENDEDD!ST$K_INFO
00000000'GF 04 FB 0851 2373          CALLS      #4,G^LIB$SIGNAL
0858 2374
0858 2375          RETURN_BLK:
0858 2376          ; now, put the used entry on the available list
0858 2377
0858 2378          INSQHI      (R11),@P_AVL_LST_HEAD(AP)      ; put entry at front of avail list
1C BC 6B 5C 0858 2379
085C 2380
085C 2381          ; if abort is set, don't touch the case
085C 2382
0C BC 0D E0 085C 2383          BBS      #TCNTRL$V_ABORT,@P_TERM_FLG(AP),-
0860 2384          END_PROCESS_TERM_SBRTN      ; don't touch case
0861 2385
0861 2386          ; now, see if there is a process ready to get started. If there
0861 2387          ; is, then set the proper flag to get it going.
0861 2388
0C BC 0C E1 0861 2389          BBC      #TCNTRL$V_PROC_PEND,@P_TERM_FLG(AP),- ; no processes are pending,
0865 2390          END_PROCESS_TERM_SBRTN      ; so leave
0866 2391
0C BC 10 BC 10 C8 0866 2392          BISL2      #CASE$M_FILE,@P_CNTRL_CASE(AP) ; try and start pending process
0C BC 00001000 8F CA 086A 2393          BICL2      #TCNTRL$M_PROC_PEND,@P_TERM_FLG(AP) ; and clear pending flag
```

TSTCNTRL
V04-000

TEST PACKAGE CONTROL PROGRAM
Process Termination Subroutine

J 5

16-SEP-1984 01:30:05
5-SEP-1984 04:26:03

VAX/VMS Macro V04-00
[UETP.SRC]UETPHAS00.MAR;1

Page 59
(32)

```
0872 2394  
0872 2395 END_PROCESS_TERM_SBRN:  
0872 2396  
50 01 D0 0872 2397          MOVL    #SS$ _NORMAL,R0          ; return success  
0875 2398  
04 0875 2399          RET
```

```
0876 2401 :  
0876 2402 : Message a record from the log file of a process we've created so that the  
0876 2403 : record is uniformly indented from the left margin, even if it contains  
0876 2404 : embedded carriage returns, line feeds and tabs.  
0876 2405 :  
0876 2406 INDENT_LOG_RECORD:  
51 19 AA D0 0876 2407 MOVL LOG_RCRD_DESC+4(R10),R1 : R1 and R0 are a string desc...  
50 15 AA 3C 087A 2408 MOVZWL LOG_RCRD_DESC(R10),R0 : ...for the remainder of the record  
7E 50 B0 087E 2409 MOVW R0,=(SP)- : Counts chars as indentation is done  
1E 11 0881 2410 BRB 30$ : BR inside loop - indent string's start  
61 50 0D 3A 0883 2411 10$: LOCC #CR,R0,(R1) : Is there a <RET> in rest of string?  
35 13 0887 2412 BEQL 40$ : Exit loop if not - no more indent  
50 D7 0889 2413 DECL R0 : Found one. LOCC has us pointing at it  
51 D6 088B 2414 INCL R1 : Point past the <RET>  
61 0A 91 088D 2415 CMPB #LF,(R1) : Is there a <LINEFEED>?  
04 12 0890 2416 BNEQ 20$ : BR if we need not skip <LINEFEED>  
50 D7 0892 2417 DECL R0 : Must pass over <LINEFEED>...  
51 D6 0894 2418 INCL R1 : ...since they're new line to printers  
0896 2419  
61 09 91 0896 2420 20$: CMPB #9,(R1) : Is there a tab at start of line?  
06 12 0899 2421 BNEQ 30$ : BR if not - we can start indenting  
50 D7 089B 2422 DECL R0 : Must pass over the tab  
51 D6 089D 2423 INCL R1 : More of passing over the tab  
F5 11 089F 2424 BRB 20$ : Inner loop to find multiple tabs  
08A1 2425  
50 D5 08A1 2426 30$: TSTL R0 : If we're at the end of the string...  
19 13 08A3 2427 BEQL 40$ : ...we can exit the outer loop  
03 BB 08A5 2428 PUSHF #M<R0,R1> : Save desc to rest of string  
04 A1 61 50 28 08A7 2429 MOVCL R0,(R1),NBR_HDR_BLNKS(R1) : Indent the rest of the string  
20 00 8F 00 2C 08AC 2430 MOVCL #0,#C,#A/ 7,- : Fill indented spaces with blanks  
04 BE 04 08B1 2431  
03 BA 08B4 2432 POPR #M<R0,R1> : Restore desc to rest of string  
51 04 C0 08B6 2433 ADDL2 #NBR_HDR_BLNKS,R1 : Point beyond the spaces just inserted  
6E 04 A0 08B9 2434 ADDW2 #NBR_HDR_BLNKS,(SP) : Count total length incl. indentation  
C5 11 08BC 2435 BRB 10$ : Loop to see if we need indent again  
08BE 2436  
15 AA 8E B0 08BE 2437 40$: MOVW (SP)+,LOG_RCRD_DESC(R10) : Set new record size  
05 08C2 2438 RSB : Return with finished record  
08C2 2439
```

```
00000004 08C3 2441 .SBTTL End-of-file Subroutine
00000008 08C3 2442
0000000C 08C3 2443 ; define offsets for parameters
08C3 2444
08C3 2445 E_CNTRL_CASE = 4
08C3 2446 EOF_FLG = 8
08C3 2447 DAT_FIL_RAB = 12
08C3 2448
08C3 2449 :++
08C3 2450 : FUNCTIONAL DESCRIPTION:
08C3 2451 :
08C3 2452 : This subroutine is called whenever an end-of-file is reached in the
08C3 2453 : TSTCNTRL data file. It determines whether or not to rewind the data
08C3 2454 : file and continue processing, or to clean up and exit.
08C3 2455 :
08C3 2456 : CALLING SEQUENCE:
08C3 2457 :
08C3 2458 : This is called when CASE$V_EOF is set by the GET_PARSE_SBRTN.
08C3 2459 : PUSHAL DATA_FILE_RAB
08C3 2460 : PUSHAL FLAGS
08C3 2461 : PUSHAL CONTROL_CASE
08C3 2462 : CALLS #3, EOF_SBRTN
08C3 2463 :
08C3 2464 : INPUT PARAMETERS:
08C3 2465 :
08C3 2466 : E_CNTRL_CASE(AP) - address of a longword holding the CONTROL_CASE flags.
08C3 2467 : EOF_FLG(AP) - address of a longword holding the TSTCNTRL flags.
08C3 2468 : DAT_FIL_RAB(AP) - address of the start of the RAB of the data file.
08C3 2469 :
08C3 2470 : IMPLICIT INPUTS:
08C3 2471 :
08C3 2472 : None.
08C3 2473 :
08C3 2474 : OUTPUT PARAMETERS:
08C3 2475 :
08C3 2476 : None.
08C3 2477 :
08C3 2478 : IMPLICIT OUTPUTS:
08C3 2479 :
08C3 2480 : The flags CASE$V_EOF, TCNTRL$V_REWOUND, CASE$V_PARSE, and
08C3 2481 : TCNTRL$V_EXIT may be altered.
08C3 2482 :
08C3 2483 : COMPLETION CODES:
08C3 2484 :
08C3 2485 : SSS_NORMAL
08C3 2486 :
08C3 2487 : SIDE EFFECTS:
08C3 2488 :
08C3 2489 : May rewind the data file.
08C3 2490 : --
08C3 2491 :
000C 08C3 2492 EOF_SBRTN:
08C3 2493 .WORD ^M<R2,R3>
08C3 2494
08C3 2495 ; turn off EOF flag so we don't come back here
08C5 2496
04 BC 08 CA 08C5 2497 BICL2 #CASE$M_EOF, @E_CNTRL_CASE(AP) ; clear EOF flag
```

				08C9	2498		
				08C9	2499		; get the address of the RAB
				08C9	2500		
53	0C	AC	D0	08C9	2501	MOV	DAT_FIL_RAB(AP),R3 ; get RAB address
				08CD	2502		
				08CD	2503		; see if data file is coming from a terminal
				08CD	2504		; first find the FAB
				08CD	2505		
52	3C	A3	D0	08CD	2506	MOV	RAB\$L_FAB(R3),R2 ; get FAB address
				08D1	2507		
				08D1	2508		; now, if data file is a terminal, then don't wrap, and time to exit
				08D1	2509		
26	40	A2	02	E0	08D1	BBS	#DEV\$V_TRM,FAB\$L_DEV(R2),10\$; terminal, so exit
				08D6	2511		
				08D6	2512		; see if we are to rewind the data file by using the following expression
				08D6	2513		; if TCNTRL\$V_ABLE_TO_WRAP ^ TCNTRL\$V_STRT_MORE ^ TCNTRL\$V_FRST_FILE_REACH
				08D6	2514		; then rewind data file.
				08D6	2515		
21	08	BC	04	E1	08D6	BBC	#TCNTRL\$V_ABLE_TO_WRAP,@EOF_FLG(AP),10\$; is wrap on?
1C	08	BC	01	E1	08DB	BBC	#TCNTRL\$V_STRT_MORE,@EOF_FLG(AP),10\$; yes it is, is start more on?
17	08	BC	00	E1	08E0	BBC	#TCNTRL\$V_FRST_FILE_REACHED,@EOF_FLG(AP),10\$; yes, have we started
				08E5	2519		; at least one process?
				08E5	2520		
				08E5	2521		; all the conditions are true, so rewind the data file
				08E5	2522		
				08E5	2523	\$REWIND	RAB = R3,- ; rewind the data file
				08E5	2524		ERR = RMSERR
				08F2	2525		
08	BC	00000400	8F	C8	08F2	BISL2	#TCNTRL\$M_REWOUND,@EOF_FLG(AP) ; and set flag saying we rewound
					08FA		
			10	11	08FA	BRB	END_EOF_SBRTN ; and leave
					08FC		
					08FC		10\$:
					08FC		; we are not to rewind the data file, so therefore it must be
					08FC		; time to leave.
					08FC		
08	BC	00000800	8F	C8	08FC	BISL2	#TCNTRL\$M_EXIT,@EOF_FLG(AP) ; time to exit now
04	BC	00000080	8F	CA	0904	BICL2	#CASE\$M_PARSE,@E_CNTRL_CASE(AP) ; don't do anymore parsing
					090C		
					090C		END_EOF_SBRTN:
					090C		
					090C		
			50	01	D0	MOV	#SS\$ _NORMAL,R0 ; success return
					090F		
				04	090F	RET	
					0910		

```
00000004 0910 2544 .SBTTL Filename found Subroutine
00000008 0910 2545
0000000C 0910 2546 ; define offsets for parameters
00000010 0910 2547
00000014 0910 2548 F_CNTRL_CASE = 4
0910 2549 F_IL_FLG = 8
0910 2550 F_PARSE_FLGS = 12
0910 2551 F_PROC_RUN = 16
0910 2552 ACT_PROC_LIM = 20
0910 2553
0910 2554 :++
0910 2555 : FUNCTIONAL DESCRIPTION:
0910 2556 :
0910 2557 : This subroutine is called when a data record with a valid filename
0910 2558 : is fetched. It determines whether or not the process can be created
0910 2559 : at this time or if we must wait for some currently running processes to
0910 2560 : end before we can create a new one.
0910 2561 :
0910 2562 : CALLING SEQUENCE:
0910 2563 :
0910 2564 : This is called when TCNTRL$V FILE is set by the GET_PARSE_SBRN.
0910 2565 : PUSHAL ACTIVE_PROC_LIMIT
0910 2566 : PUSHAL PROCESS_RUNNING
0910 2567 : PUSHAL PARSE_FLAGS
0910 2568 : PUSHAL FLAGS
0910 2569 : PUSHAL CONTROL_CASE
0910 2570 : CALLS #5,FILE_SBRN
0910 2571 :
0910 2572 : INPUT PARAMETERS:
0910 2573 :
0910 2574 : F_CNTRL_CASE(AP) - address of a longword holding the CONTROL_CASE flags.
0910 2575 : F_IL_FLG(AP) - address of a longword holding the TSTCNTRL flags.
0910 2576 : F_PARSE_FLGS(AP) - address of a longword holding the PARSE_FLAGS.
0910 2577 : F_PROC_RUN(AP) - address of a longword holding a count of the number
0910 2578 : of processes currently running.
0910 2579 : ACT_PROC_LIM(AP) - address of a longword holding a value of the number
0910 2580 : of processes allowed to be active at the same time.
0910 2581 :
0910 2582 : IMPLICIT INPUTS:
0910 2583 :
0910 2584 : None.
0910 2585 :
0910 2586 : OUTPUT PARAMETERS:
0910 2587 :
0910 2588 : None.
0910 2589 :
0910 2590 : IMPLICIT OUTPUTS:
0910 2591 :
0910 2592 : The flags TCNTRL$V FILE, TCNTRL$V_PARSE, TCNTRL$V_CREATE_PROC,
0910 2593 : and TCNTRL$V_PROC_PEND may be altered.
0910 2594 :
0910 2595 : COMPLETION CODES:
0910 2596 :
0910 2597 : SSS_NORMAL
0910 2598 :
0910 2599 : SIDE EFFECTS:
0910 2600 :
```

```
0910 2601 :-- None.
0910 2602 :--
0910 2603
0910 2604 FILE_SBRTN:
0000 0910 2605 .WORD ^M<>
0912 2606
0912 2607 ; turn off file flag and parse flag so that we don't go there when
0912 2608 ; through here.
0912 2609
04 BC 04 BC 10 CA 0912 2610 BICL2 #CASE$M_FILE,@F_CNTRL_CASE(AP) ; clear FILE flag
04 BC 00000080 8F CA 0916 2611 BICL2 #CASE$M_PARSE,@F_CNTRL_CASE(AP) ; clear GET_PARSE flag
091E 2612
091E 2613 ; now, see if this file is to be run by itself and if the number of
091E 2614 ; current processes is zero
091E 2615
00 00 E1 091E 2616 BBC #PARSE$V_PROC_RUNS_OTHERS,- ; if can't run with others
0C BC 02 0920 2617 @F_PARSE_FLG$AP,= ; then process must be started
0922 2618 SEQUENTIAL_PROC ; by itself
0923 2619
20 11 0923 2620 BRB CONCURRENT_PROC ; process can run with others
0925 2621
0925 2622 SEQUENTIAL_PROC:
0925 2623 ; This process does run alone, now see if the number of running
0925 2624 ; processes is zero
0925 2625
10 10 CA 0925 2626 BICL2 #TCNTRL$M_ABLE_TO_WRAP,- ; process runs sequentially,
08 BC 08 BC 0927 2627 @FIL_FLG(AP) ; cannot wrap
0929 2628
10 BC 00 D1 0929 2629 CMPL #0,@F_PROC_RUN(AP) ; are any processes running?
08 13 092D 2630 BEQL 10$ ; yes, so don't start this one yet
092F 2631
092F 2632 ; other processes are currently running, so make this one pending to
092F 2633 ; be run
092F 2634
08 BC 00001000 8F C8 092F 2635 BISL2 #TCNTRL$M_PROC_PEND,@FIL_FLG(AP) ; set process pending
0024 31 0937 2636 BRW END_FILE_SBRTN ; and leave
093A 2637
093A 2638 10$:
093A 2639 ; no processes are running, so we can start this one up
093A 2640
04 BC 00000040 8F C8 093A 2641 BISL2 #CASE$M_CREATE_PROC,@F_CNTRL_CASE(AP) ; signal to start this proces
0019 31 0942 2642 BRW END_FILE_SBRTN ; and do it
0945 2643
0945 2644
0945 2645 CONCURRENT_PROC:
0945 2646 ; We couldn't start this process yet because it wasn't running
0945 2647 ; by itself.
0945 2648 ; since this process can run with others, see if the number
0945 2649 ; of currently running processes is less than the maximum number
0945 2650 ; of processes which are allowed to run together.
0945 2651
0945 2652 ; is the number of processes running less than the maximum?
0945 2653
14 BC 10 BC D1 0945 2654 CMPL @F_PROC_RUN(AP),@ACT_PROC_LIM(AP) ; see if we can start this one
0A 18 094A 2655 BGEQ 30$ ; we can't so make it pending
094C 2656
094C 2657 ; we can start this process so signal that
```

```
04 BC 00000040 8F C8 094C 2658
      08 11 094C 2659 BISL2 #CASE$M_CREATE_PROC,@F_CNTRL_CASE(AP) ; we can start this one
      0954 2660 BRB END_FILE_SBRTN ; so do it
      0956 2661
      0956 2662 30$:
      0956 2663 ; we cannot start this process yet, so make it pending
      0956 2664
08 BC 00001000 8F C8 0956 2665 BISL2 #TCNTRL$M_PROC_PEND,@FIL_FLG(AP) ; pending process
      095E 2666
      095E 2667 END_FILE_SBRTN:
      095E 2668
      50 01 D0 095E 2669 MOVL #SS$ _NORMAL,R0 ; success return
      0961 2670
      04 0961 2671 RET
```

```
0962 2673      .SBTTL Create Process Subroutine
0962 2674
0962 2675      ; define offsets for parameters
0962 2676
00000004 0962 2677      TOT_PROC_RUN = 4
00000008 0962 2678      C_PROC_RUN = 8
0000000C 0962 2679      CRT_PRC_FLG = 12
00000010 0962 2680      C_CNTRL_CASE = 16
00000014 0962 2681      CRT_PRC_CASE = 20
00000018 0962 2682      AVL_LST_HEAD = 24
0000001C 0962 2683      CURR_LST_HEAD = 28
00000020 0962 2684      C_TOT_PRC_LIM = 32
00000024 0962 2685      C_PARSE_FLAGS = 36
0962 2686
0962 2687      ++
0962 2688      : FUNCTIONAL DESCRIPTION:
0962 2689      :
0962 2690      : This subroutine is called when a process is ready to be created;
0962 2691      : ie., after the filespec has been parsed out completely and after
0962 2692      : the number of active processes is at a proper value. The routine
0962 2693      : GET_PARSE will parse out the filename and will also determine what
0962 2694      : type of file it is to be started (it communicates this by setting
0962 2695      : a flag in CREATE_PROC_CASE).
0962 2696      : This routine will then perform all the actions required to start this
0962 2697      : process and handle any problems caused by the process not activating.
0962 2698
0962 2699      : CALLING SEQUENCE:
0962 2700      :
0962 2701      : This routine is called when CASE$V_CREATE_PROC is set by FILE_SBRTN
0962 2702      : PUSHAL PARSE_FLAGS
0962 2703      : PUSHAL TOTAL_PROC_LIMIT
0962 2704      : PUSHAQ CURRENT_LIST_HEAD
0962 2705      : PUSHAQ AVAIL_LIST_HEAD
0962 2706      : PUSHAL CREATE_PROC_CASE
0962 2707      : PUSHAL CONTROL_CASE
0962 2708      : PUSHAL FLAGS
0962 2709      : PUSHAL PROCESS_RUNNING
0962 2710      : PUSHAL TOTAL_PROC_RUN
0962 2711      : CALLS #9.CREATE_PROC_SBRTN
0962 2712
0962 2713      : INPUT PARAMETERS:
0962 2714      :
0962 2715      : TOT_PROC_RUN(AP) - address of the cumulative number of total processes
0962 2716      : run this session
0962 2717      : C_PROC_RUN(AP) - address of the number of processes currently running
0962 2718      : CRT_PRC_FLG(AP) - address of the TSTCNTRL flags
0962 2719      : C_CNTRL_CASE(AP) - address of the CONTROL_CASE longword
0962 2720      : CRT_PRC_CASE(AP) - address of the CREATE_PROC_CASE longword (initialized
0962 2721      : by GET_PARSE_SBRTN)
0962 2722      : AVL_LST_HEAD(AP) - quadword header of the list of available slots
0962 2723      : CURR_LST_HEAD(AP) - quadword header of the list of processes which are
0962 2724      : currently active
0962 2725      : C_TOT_PRC_LIM(AP) - address of the number of allowed detached processes.
0962 2726      : C_PARSE_FLAGS(AP) - address of the flags resulting from the parse routine
0962 2727
0962 2728      : IMPLICIT INPUTS:
0962 2729
```

```
0962 2730 : The parsed file specification is implicit input to this routine.
0962 2731 : This is made by GET_PARSE_SBRTN and is a contiguous string with
0962 2732 : descriptors for the various pieces of the specification. Therefore,
0962 2733 : the following descriptors are implicit inputs (again, note that these
0962 2734 : get set by GET_PARSE_SBRTN):
0962 2735 :
0962 2736 :     FILE_SPEC_DESC - descriptor for the entire file specification
0962 2737 :     NODE_DESC      - descriptor for the node on which the file resides
0962 2738 :     DEVICE_DESC    - descriptor for the device designation
0962 2739 :     DIRECTORY_DESC - descriptor for the directory of the file
0962 2740 :     FILENAME_DESC  - descriptor of the actual filename
0962 2741 :     TYPE_DESC      - descriptor of the extension of the file
0962 2742 :     VERSION_DESC   - descriptor of the files version number
0962 2743 :     PARAM_DESC     - descriptor of the parameters associated with the
0962 2744 :                     file (if there are any)
0962 2745 :     COMMENT_DESC   - descriptor of the comment field from the data
0962 2746 :                     file (if any)
0962 2747 :
0962 2748 : Other information as needed in order to start detached process.
0962 2749 :     UIC_CODE - to be UIC of the detached process
0962 2750 :     MBX_UNIT - the mailbox unit number for the termination mailbox
0962 2751 :
0962 2752 : OUTPUT PARAMETERS:
0962 2753 :
0962 2754 :     None
0962 2755 :
0962 2756 : IMPLICIT OUTPUTS:
0962 2757 :
0962 2758 :     Implicit outputs include filling in the values of the $CREPRC input
0962 2759 :     list. The values which are the same for all processes created are
0962 2760 :     the following:
0962 2761 :         PIDADR - points to where the PID of the created process is
0962 2762 :                 to be placed
0962 2763 :         OUTPUT - points to what SYS$OUTPUT is (the temporary log file)
0962 2764 :         ERROR  - points to what SYS$ERROR is (the temporary log file)
0962 2765 :         PRCNAM - points to what the process name is
0962 2766 :         UIC    - points to what the UIC is to be
0962 2767 :         MBXUNT - points to the mailbox unit number of the termination
0962 2768 :                 mailbox
0962 2769 :
0962 2770 : COMPLETION CODES:
0962 2771 :
0962 2772 :     SSS_NORMAL
0962 2773 :
0962 2774 : SIDE EFFECTS:
0962 2775 :
0962 2776 :     May create a detached process.
0962 2777 :
0962 2778 : --
0962 2779 :
0962 2780 : CREATE_PROC_SBRTN:
OFFC 0962 2781 :     .WORD ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
0964 2782 :
0964 2783 :
0964 2784 : ; set the control flags so that we don't come back here, and that
0964 2785 : ; we go to the parse routine
0964 2786 :
```

10 BC	00000040	8F	CA	0964	2787	BICL2	#CASE\$M_CREATE_PROC,@C_CNTRL_CASE(AP)	; don't come back				
10 BC	00000080	8F	C8	096C	2788	BISL2	#CASE\$M_PARSE,@C_CNTRL_CASE(AP)	; but do parse instead				
		01	C8	0974	2789							
		0C BC		0974	2790	BISL2	#TCNTRL\$M_FRST_FILE_REACHED,-	; we've started at least one				
				0976	2791		@CRT_PRC_FLG(AP)	; process				
				0978	2792							
				0978	2793			; if we have a null file, don't do anything. go right to case.				
				0978	2794							
03 14 BC	04		E1	0978	2795	BBC	#CRTPRC\$V_NULL,@CRT_PRC_CASE(AP),10\$	; if real file, continue				
	0124		31	097D	2796	BRW	CASE_FILE_TYPE	; if null, then case				
				0980	2797							
				0980	2798							
				0980	2799			10\$: ; now initialize the stack area for local storage				
				0980	2800							
5E	00000045	8F	C2	0980	2801	SUBL2	#CREATE_PROC_LENGTH,SP	; allocate space for local storage				
				0987	2802							
6E	0045	8F	00	00	8F	00	2C	0987	2803	MOVCS	#0,#0,#0,#CREATE_PROC_LENGTH,(SP)	; clear out local storage
		5B	5E				D0	0990	2804	MOVL	SP,R11	; keep pointer to local storage
								0993	2805			
								0993	2806			
								0993	2807			; now initialize the descriptors in the local area
								0993	2808			
								0993	2809			; first, the descriptor for the process name
								0993	2810			
	08 AB		DE	0993	2811	MOVAL	TEMP_PNAM_STR(R11),-					
	04 AB			0996	2812		<TEMP_PNAM_DESC+DSC\$A_POINTER>(R11)	; process name descriptor				
				0998	2813							
				0998	2814			; then the descriptor for log file of the process				
				0998	2815							
	1F AB		DE	0998	2816	MOVAL	TEMP_LOG_STR(R11),-					
	1B AB			0998	2817		<TEMP_LOG_DESC+DSC\$A_POINTER>(R11)	; log file name				
				099D	2818							
				099D	2819			; now, make the process name				
				099D	2820							
	089C'CF		7F	099D	2821	PUSHAQ	FILENAME_DESC	; the test filename				
	04 AC		DD	09A1	2822	PUSHL	TOT_PROC_RUN(AP)	; the ID number				
			7F	09A4	2823	PUSHAQ	TEMP_PNAM_DESC(R11)	; the process name				
ODC7'CF	03		FB	09A6	2824	CALLS	#3,MAKE_PROC_NAM	; now, make the process name				
				09AB	2825							
				09AB	2826			; make the temporary log file name				
				09AB	2827							
	089C'CF		7F	09AB	2828	PUSHAQ	FILENAME_DESC	; the test filename				
	04 AC		DD	09AF	2829	PUSHL	TOT_PROC_RUN(AP)	; the ID number				
	041C'CF		7F	09B2	2830	PUSHAQ	LOG_EXT_DESC	; the temp log file extension				
	17 AB		7F	09B6	2831	PUSHAQ	TEMP_LOG_DESC(R11)	; the temp. log file				
OE35'CF	04		FB	09B9	2832	CALLS	#4,MAKE_TMP_LOG_NAM	; and make the log file name				
				09BE	2833							
				09BE	2834							
				09BE	2835			; get a slot off of the available list				
				09BE	2836							
5A	18 BC		5E	09BE	2837	REMOHL	@AVL_LST_HEAD(AP),R10	; address of free slot				
				09C2	2838							
	25		1C	09C2	2839	BVC	30\$	; we got the slot, now fill it in				
				09C4	2840							
				09C4	2841			; the available queue is empty, allocate and initialize more slots and				
				09C4	2842			; try again				
				09C4	2843							

```
09C4 2844 ; first allocate more space
09C4 2845
09C4 2846
00000168 3D AB 7F DD 09C7 2847 PUSHAQ STRCT TOP BOT(R11) ; top & bottom of new structure
00000168 8F DD 09C7 2847 PUSHL #PRCINFO$ PRC_INFO ; byte count of the slot size
00000168 32 DD 09CD 2848 PUSHL #NUMB OF SLOTS ; number of slots we want
136B'CF 03 FB 09CF 2849 CALLS # FXPND_REGION ; get more space
09D4 2850
09D4 2851 ; now initialize the space we've just acquired
09D4 2852
09D4 2853
09D4 2854 PUSHAQ @AVL_LST HEAD(AP) ; the available list pointer
09D7 2854 PUSHL STRCT TOP BOT(R11) ; top of new structure
00000168 3D AB DD 09DA 2855 PUSHL #PRCINFO$ PRC_INFO ; byte count of the slot size
00000168 8F DD 09E0 2856 PUSHL #NUMB OF SLOTS ; number of slots we want
134C'CF 04 FB 09E2 2857 CALLS #4,INIT_STRUCT ; init the structure
09E7 2858
09E7 2859 BRB 20$ ; now try and get a slot again
09E9 2860
09E9 2861 30$:
09E9 2862 ; now fill the new slot with as much info as possible
09E9 2863
09E9 2864 ; first fill in the parsed file information. Move the entire filespec
09E9 2865 ; to the new slot
09E9 2866
09E9 2867 MOV C3 <FILE_SPEC_DESC+DSC$W_LENGTH>,- ; source length
077D'CF 28 09ED 2868 @<FILE_SPEC_DESC+DSC$A_POINTER>,- ; source address
0781'DF 09F0 2869 PRCINFO$ FILE_SPEC(R10) ; put it in the new slot
6C AA 09F2 2870
09F2 2871
09F2 2872 ; now fill in the sizes and the pointers of the parsed filespec
09F2 2873 MOVAL PRCINFO$ FILE_SPEC(R10),R5 ; get the address of the start
55 6C AA DE 09F6 2874 ; of the file spec
09F6 2875
09F6 2876 ; first the overall filespec
09F6 2877
09F6 2878 MOVW <FILE_SPEC_DESC+DSC$W_LENGTH>,- ; move the length to the
077D'CF 80 09FA 2879 PRCINFO$ FILESPEC_SIZ(R10) ; new slot
58 AA 09FC 2880
09FC 2881
09FC 2882 ; next do the node name
09FC 2883
09FC 2884 MOVW <NODE_DESC+DSC$W_LENGTH>,- ; move the length to the
0884'CF 80 0A00 2885 PRCINFO$ NODE_SIZ(R10) ; place for the node size
5A AA 0A02 2886
0A02 2887
0A02 2888 MOVL R5,PRCINFO$A_NODE(R10) ; and put it in the slot
40 AA 55 D0 0A06 2889
0A06 2888 MOVZWL PRCINFO$ NODE_SIZ(R10),R4 ; clear upper word of length
54 5A AA 3C 0A0A 2889
0A0A 2890
0A0A 2891 ADDL2 R4,R5 ; adjust the address to be
55 54 C0 0A0D 2892 ; the device
0A0D 2893
0A0D 2894 ; then the device
0A0D 2895
0A0D 2896 MOVW <DEVICE_DESC+DSC$W_LENGTH>,- ; move the length to the
088C'CF 80 0A11 2897 PRCINFO$ DEVICE_SIZ(R10) ; place for the device size
5C AA 0A13 2898
0A13 2899
0A13 2898 MOVL R5,PRCINFO$A_DEVICE(R10) ; and put it in the slot
44 AA 55 D0 0A17 2899
0A17 2900
0A17 2900 MOVZWL PRCINFO$ DEVICE_SIZ(R10),R4 ; clear upper word of length
```

55	54	C0	0A1B	2901			
			0A1B	2902	ADDL2	R4,R5	; adjust the address to be
			0A1E	2903			; the directory
			0A1E	2904			
			0A1E	2905			; then the directory
			0A1E	2906			
0894'	CF	B0	0A1E	2907	MOVW	<DIRECTORY_DESC+DSC\$W_LENGTH>,-	; move the length to the
5E	AA		0A22	2908		PRCINFO\$W_DIR_SIZ(R10)	; place for directory size
			0A24	2909			
44	AA	55	0A24	2910	MOVL	R5,PRCINFO\$A_DEVICE(R10)	; and put it in the slot
			0A28	2911			
54	5E	AA	0A28	2912	MOVZWL	PRCINFO\$W_DIR_SIZ(R10),R4	; clear upper word of length
			0A2C	2913			
55	54	C0	0A2C	2914	ADDL2	R4,R5	; adjust the address to be the
			0A2F	2915			; filename
			0A2F	2916			
			0A2F	2917			; then the filename
			0A2F	2918			
089C'	CF	B0	0A2F	2919	MOVW	<FILENAME_DESC+DSC\$W_LENGTH>,-	; move the length to the
60	AA		0A33	2920		PRCINFO\$W_FILNAM_SIZ(R10)	; place for the filename size
			0A35	2921			
4C	AA	55	0A35	2922	MOVL	R5,PRCINFO\$A_FILNAM(R10)	; and put it in the slot
			0A39	2923			
54	60	AA	0A39	2924	MOVZWL	PRCINFO\$W_FILNAM_SIZ(R10),R4	; clear upper word of length
			0A3D	2925			
55	54	C0	0A3D	2926	ADDL2	R4,R5	; adjust the address to be the
			0A40	2927			; file extension
			0A40	2928			
			0A40	2929			; then the file extension
			0A40	2930			
08A4'	CF	B0	0A40	2931	MOVW	<TYPE_DESC+DSC\$W_LENGTH>,-	; move the length to the
62	AA		0A44	2932		PRCINFO\$W_EXTENSION_SIZ(R10)	; place for the extension size
			0A46	2933			
50	AA	55	0A46	2934	MOVL	R5,PRCINFO\$A_EXTENSION(R10)	; and put it in the slot
			0A4A	2935			
54	62	AA	0A4A	2936	MOVZWL	PRCINFO\$W_EXTENSION_SIZ(R10),R4	; clear upper word of length
			0A4E	2937			
55	54	C0	0A4E	2938	ADDL2	R4,R5	; adjust the address to be the
			0A51	2939			; version number
			0A51	2940			
			0A51	2941			; and finally the version number
			0A51	2942			
08AC'	CF	B0	0A51	2943	MOVW	<VERSION_DESC+DSC\$W_LENGTH>,-	; move the length to the
64	AA		0A55	2944		PRCINFO\$W_VERSION_SIZ(R10)	; place for the version size
			0A57	2945			
54	AA	55	0A57	2946	MOVL	R5,PRCINFO\$A_VERSION(R10)	; and put it in the slot
			0A5B	2947			
			0A5B	2948			; now, put the ID number in the process info slot
			0A5B	2949			
3E	AA	04	0A5B	2950	MOVW	@TOT_PROC_RUN(AP),PRCINFO\$W_ID_NUMB(R10)	; put the ID number in slot
			0A60	2951			
			0A60	2952			
			0A60	2953			; make sure all the flags are cleared
			0A60	2954			
38	AA	D4	0A60	2955	CLRL	PRCINFO\$L_FLAGS(R10)	; clear the flags
			0A63	2956			
			0A63	2957			; before creating a process, delete any left-over log files

```
0A63 2958
0A63 2959 $FAB_STORE FAB = TMP_LOG_FAB,- ; put in the name of current log fil
0A63 2960 FNA = TEMP_LOG_STR(R11),-
0A63 2961 FNS = TEMP_LOG_DESC(R11),-
0A72 2962 $ERASE FAB = TMP_LOG_FAB,- ; erase the log file
0A72 2963 ERR = RMSERR
0A81 2964
0A81 2965 ; set up generic parameter list for $CREPRC
0A8 2966
042E'CF 0C AA DE 0A81 2967 MOVAL PRCINFO$ PID(R10),PIDADR ; get the PID
043A'CF 17 AB 7E 0A87 2968 MOVAQ TEMP_LOG_DESC(R11),OUTPUT ; and what SYSS$OUTPUT is
044A'CF 6B 7E 0A8D 2969 MOVAQ TEMP_PNAM_DESC(R11),PRCNAM ; and the process name
0452'CF 04FA'CF D0 0A92 2970 MOVL UIC_CODE,UIC ; and the UIC
0456'CF 0428'CF 3C 0A99 2971 MOVZWL MBX_UNIT,MBXUNT ; don't forget the mailbox
0AA0 2972
0AA0 2973 ; now assume that the process will be created successfully and
0AA0 2974 ; put the slot on the current_running list.
0AA0 2975
1C BC 6A 5D 0AA0 2976 INSQTI (R10),@CURR_LST_HEAD(AP) ; put slot on curr. run. list
0AA4 2977
0AA4 2978 CASE_FILE_TYPE:
0AA4 2979 ; now, we will start up the process. The GET_PARSE routine has
0AA4 2980 ; determined what type of process we are trying to start up, so
0AA4 2981 ; we will case on the results of that.
0AA4 2982
00 EA 0AA4 2983 FFS #0,- ; case off of flag, start at zero
20 0AA6 2984 #<CRTPRC$S CRT_PROC_CASE*8>,- ; go this many bits
14 BC 0AA7 2985 @CRT_PRC_CASE(AP),- ; this holds the flags
59 0AA9 2986 R9 ; and put the results here
0AAA 2987
04 00 59 8F 0AAA 2988 CASEB R9,#0,#<CRTPRC$K_CREAT_CASE_SIZE-1> ; perform case
000D' 0AAE 2989 10$: .WORD EXE_FILE - 10$
0015' 0AB0 2990 .WORD EXE_PARM - 10$
003A' 0AB2 2991 .WORD COM_FILE - 10$
0042' 0AB4 2992 .WORD COM_PARM - 10$
000A' 0AB6 2993 .WORD NULL_FILE - 10$
0AB8 2994
0AB8 2995
0AB8 2996 NULL_FILE:
0AB8 2997 ; can come here either by the proper bit being set, or by falling
0AB8 2998 ; through the case. If we fall through the case (an error in the
0AB8 2999 ; the filespec) handle it as if a null file. With a null file,
0AB8 3000 ; don't do any processing, just exit. This is used mainly to
0AB8 3001 ; to run down currently running processes (done by running a null
0AB8 3002 ; file sequentially).
0AB8 3003
010A 31 0AB8 3004 BRW END_CREATE_PROC
0ABB 3005
0ABB 3006
0ABB 3007 EXE_FILE:
0ABB 3008 ; reached by the filespec being an executable image, and by having
0ABB 3009 ; no parameters.
0ABB 3010
0BC9'CF 00 FB 0ABB 3011 CALLS #0,CRT_EXE_PRC ; create the process
0AC0 3012
0035 31 0AC0 3013 BRW CHCK_STATUS ; see if finished without error
0AC3 3014
```

```

OAC3 3015 EXE_PARM:
OAC3 3016 ; reached by the filespec being an executable image, and by having
OAC3 3017 ; parameters
OAC3 3018
OAC3 3019 ; first make a unique logical name to be associated with the physical
OAC3 3020 ; name of the mailbox
OAC3 3021
34 AB DE OAC3 3022 MOVAL MBX_LOGNAM_STR(R11),- ; initialize the descriptor
30 AB OAC6 3023 <MBX_LOGNAM_DESC+DSC$A_POINTER>(R11)
OAC8 3024
OAC8 3025 ; and make a unique name (which is the same as the temp log file
OAC8 3026 ; name expect is doesn't have an extension).
OAC8 3027
OAC8 3028 PUSHAQ FILENAME_DESC ; use the name of the file
04 AC DD OACC 3029 PUSHL TOT_PROC_RUN(AP) ; along with the ID number
0428'CF 7F OACF 3030 PUSHAQ NIL_EXT_DESC ; and no extension
2C AB 7F OAD3 3031 PUSHAQ MBX_LOGNAM_DESC(R11) ; and put results here
OE35'CF 04 FB OAD6 3032 CALLS #4,MAKE_TMP_LOG_NAM ; make the logical name
OADB 3033
5A DD OADB 3034 PUSHL R10 ; process info slot
2C AB 7F OADD 3035 PUSHAQ MBX_LOGNAM_DESC(R11) ; the logical name
OBE6'CF 02 FB OAE0 3036 CALLS #2,CRT_EXE_PARM ; create the process
OAE5 3037
0010 31 OAE5 3038 BRW CHCK_STATUS ; see if finished without error
OAE8 3039
OAE8 3040 COM_FILE:
OAE8 3041 ; reached by the filespec being a command procedure, having no
OAE8 3042 ; parameters
OAE8 3043
OC47'CF 00 FB OAE8 3044 CALLS #0,CRT_COM_PRC ; create the process
OAE8 3045
0008 31 OAE8 3046 BRW CHCK_STATUS ; see if finished without error
OAF0 3047
OAF0 3048 COM_PARM:
OAF0 3049 ; reached by the filespec being a command procedure with parameters
OAF0 3050
OAF0 3051 PUSHL TOT_PROC_RUN(AP) ; total processes run thus far
OC6E'CF 01 FB OAF3 3052 CALLS #1,CRT_COM_PARM ; create the process
OAF8 3053
OAF8 3054 CHCK_STATUS:
OAF8 3055 ; we've attempted to create the detached processes, if for some reason
OAF8 3056 ; we did not succeed, then print out a warning message saying why.
OAF8 3057
5F 50 E8 OAF8 3058 BLBS R0,INC_COUNT ; Go increment count if no err
OAFB 3059
OAFB 3060 ; there was an error, so print it
OAFB 3061 ; make the output message
OAFB 3062
55 50 D0 OAFB 3063 MOVL R0,R5 ; save the error status
OAFE 3064
OAFE 3065 ; since the process did not get started, remove the slot from the
OAFE 3066 ; current running list . . .
OAFE 3067
5A 1C BC 5F OAFE 3068 REMQTI @CURR_LST_HEAD(AP),R10 ; remove from current running list
O802 3069
O802 3070 ; . . . and then return the slot to the available list
O802 3071
```

```
18 BC 6A 5C 0B02 3072 INSQHI (R10),@AVL_LST_HEAD(AP) ; put at head of available list
0B06 3073
0B06 3074 $FAO_S CTRSTR = NO CRT_PRC,- ; make a string with this message
0B06 3075 OUTLEN = FAO_BUF,- ; get size of buffer
0B06 3076 OUTBUF = BUFFER_PTR,- ; put output here
0B06 3077 P1 = @TEMP_PNAM_DESC(R11) ; this is the process name
0B1C 3078
0B1C 3079 $GETMSG_S MSGID = R5,- ; get the message that goes with
0B1C 3080 MSGLEN = GETMSG_PTR,- ; this message
0B1C 3081 BUFADR = GETMSG_DESC,-
0B1C 3082 FLAGS = #15,-
0B1C 3083 OUTADR = MSG_BLOCK
05C9'CF DF 0B33 3084 PUSHAL GETMSG_PTR ; Push the desc address
08 DD 0B37 3085 PUSHL #8
00010002 8F DD 0B39 3086 PUSHL #^X10002
007480B8 8F DD 0B3F 3087 PUSHL #UETP$ COPY_LOG_LINE-STSS$K_SUCCESS+STSS$K_WARNING
052D'CF DF 0B45 3088 PUSHAL FAO_BUF ; send the error message
01 DD 0B49 3089 PUSHL #1
00C31130 8F DD 0B4B 3090 PUSHL #TCNTRL$ TEXT!STSS$K_WARNING
00000000'GF 07 FB 0B51 3091 CALLS #7,G^LIB$SIGNAL
36 11 0B58 3092 BRB INC_TOTAL_RUN ; increment total number run
0B5A 3093
0B5A 3094 INC_COUNT:
0B5A 3095 ; now, increment the counters that we keep. Increment both the
0B5A 3096 ; count of active processes, and the count of total processes run
0B5A 3097
08 BC D6 0B5A 3098 INCL @C_PROC_RUN(AP) ; one more process running
0B5D 3099
0B5D 3100 ; print the begin time stamp for detached process
0B5D 3101
0C BC 20 CA 0B5D 3102 BICL2 #TCNTRL$M_WRT_MSG,@CRT_PRC_FLG(AP) ; write only if long
7E D4 0B61 3103 CLRL -(SP) ; set up for begin time stamp
68 DF 0B63 3104 PUSHAL TEMP_PNAM_DESC(R11)
02 DD 0B65 3105 PUSHL #2
00C3103B 8F DD 0B67 3106 PUSHL #TCNTRL$ BEGIND!STSS$K_INFO
00000000'GF 04 FB 0B6D 3107 CALLS #4,G^LIB$SIGNAL
0B74 3108
0B74 3109 ;print out the comments?
0B74 3110
0C BC 06 E1 0B74 3111 BBC #TCNTRL$V_PRNT_COMMENTS,@CRT_PRC_FLG(AP),-
17 0B78 3112 INC_TOTAL_RUN ; no comments wanted
0C BC 20 CA 0B79 3113 BICL2 #TCNTRL$M_WRT_MSG,@CRT_PRC_FLG(AP) ; write only if long
06E9'CF DF 0B7D 3114 PUSHAL COMMENT_DESC
01 DD 0B81 3115 PUSHL #1
00C31133 8F DD 0B83 3116 PUSHL #TCNTRL$ TEXT!STSS$K_INFO
00000000'GF 03 FB 0B89 3117 CALLS #3,G^LIB$SIGNAL ; print comment
0B90 3118
0B90 3119 INC_TOTAL_RUN:
0B90 3120 ; this is the number of total processes run. If there was an error
0B90 3121 ; while trying to start up the detached process, we will only increment
0B90 3122 ; this value. This is so that the TSTCNTRL will stop in some finite
0B90 3123 ; amount of time, even though no processes can be started.
0B90 3124
04 BC D6 0B90 3125 INCL @TOT_PROC_RUN(AP) ; and one more to the total
0B93 3126
0B93 3127 ; if the process we just started was to be run alone, then don't
0B93 3128 ; parse anything until it stops
```

```

      24 BC 00 E0 0B93 3129
      08      0B93 3130
10 BC 00000080 8F CA 0B97 3131
      0B98 3132
      0BA0 3133
      0BA0 3134 10$:
      0BA0 3135
      0BA0 3136
      0BA0 3137
      20 BC 04 BC D1 0BA0 3138
      1E 19 0BA5 3139
      0BA7 3140
      0BA7 3141
      0BA7 3142
      0BA7 3143
      0C BC 02 CA 0BA7 3144
      0BAB 3145
      0BAB 3146
      0BAB 3147
      0BAB 3148
      0C BC 04 E1 0BAB 3149
      15 0BAF 3150
      0C BC 0A E1 0BB0 3151
      10 0BB4 3152
      0BB5 3153
      0BB5 3154
      0BB5 3155
      0BB5 3156
      0C BC 00000800 8F C8 0BB5 3157
      10 BC 00000080 8F CA 0BB0 3158
      0BC5 3159
      0BC5 3160 END_CREATE_PROC:
      0BC5 3161
      50 01 D0 0BC5 3162
      0BC8 3163
      04 0BC8 3164

      BBS #PARSE$V_PROC_RUNS_OTHERS,@C_PARSE_FLGS(AP),- ; if runs with
      10$ ; others, then continue
      BICL2 #CASE$M_PARSE,@C_CNTRL_CASE(AP) ; otherwise, don't parse until done

      ; now see if we should start up more processes, or if the one just
      ; started was our last one.
      CMPL @TOT_PROC_RUN(AP),@C_TOT_PRC_LIM(AP) ; have we reached our limit?
      BLSS END_CREATE_PROC ; no, start more up

      ; we have started the requested number of processes, so don't start
      ; any more
      BICL2 #TCNTRL$M_STRT_MORE,@CRT_PRC_FLG(AP) ; don't start any more

      ; now see if it is time to exit (check ~STRT_MORE ^ ABLE_TO_WRAP ^
      ; REWOUND)
      BBC #TCNTRL$V_ABLE_TO_WRAP,@CRT_PRC_FLG(AP),- ; have we been able
      END_CREATE_PROC ; to wrap around?

      BBC #TCNTRL$V_REWOUND,@CRT_PRC_FLG(AP),- ; has the data file
      END_CREATE_PROC ; been rewound?

      ; if the above expression is true, then we are to exit now!
      BISL2 #TCNTRL$M_EXIT,@CRT_PRC_FLG(AP) ; exit the TSTCNTRL
      BICL2 #CASE$M_PARSE,@C_CNTRL_CASE(AP) ; don't parse any more

      MOVL #SS$NORMAL,R0 ; return completion code
      RET
```

```
0BC9 3166 .SBTTL Create image procedure
0BC9 3167 :++
0BC9 3168 : FUNCTIONAL DESCRIPTION:
0BC9 3169 :
0BC9 3170 : This routine is called when a executable image is to be run as a
0BC9 3171 : detached process. The routine is started by running the image as
0BC9 3172 : a detached process.
0BC9 3173 :
0BC9 3174 : CALLING SEQUENCE:
0BC9 3175 :
0BC9 3176 : CALLS #0,CRT_EXE_PRC
0BC9 3177 :
0BC9 3178 : INPUT PARAMETERS:
0BC9 3179 :
0BC9 3180 : None
0BC9 3181 :
0BC9 3182 : IMPLICIT INPUTS:
0BC9 3183 :
0BC9 3184 : File specification information from GET_PARSE:
0BC9 3185 : FILE_SPEC_DESC - descriptor of the entire file specification
0BC9 3186 :
0BC9 3187 : OUTPUT PARAMETERS:
0BC9 3188 :
0BC9 3189 : None.
0BC9 3190 :
0BC9 3191 : IMPLICIT OUTPUTS:
0BC9 3192 :
0BC9 3193 : None
0BC9 3194 :
0BC9 3195 : COMPLETION CODES:
0BC9 3196 :
0BC9 3197 : SSS_NORMAL
0BC9 3198 : Any completion code which may be produced by $CREPRC
0BC9 3199 :
0BC9 3200 : SIDE EFFECTS:
0BC9 3201 :
0BC9 3202 : A detached process may be created
0BC9 3203 :--
0BC9 3204 :
0000 0BC9 3205 CRT_EXE_PRC:
0BC9 3206 .WORD ^M<>
0BCB 3207
0BCB 3208 ; set up generic parameters for executable images
0BCB 3209
0436'CF 00 D0 0BCB 3210 MOVL #0,INPUT ; there is no input for an image
045A'CF 00 D0 0BD0 3211 MOVL #0,STSFLG ; clear the status flag
0BD5 3212
0BD5 3213 ; put the name of the image in the parameter list
0BD5 3214
0432'CF 077D'CF 7E 0BD5 3215 MOVAQ FILE_SPEC_DESC,IMAGE ; run the actual image
0BDC 3216
0BDC 3217 ; and create the process
0BDC 3218
0BDC 3219 $CREPRC_G CREPRC_LIS ; create the process
0BE5 3220
04 0BE5 3221 RET ; return with status in R0
```

```
00000004 OBE6 3223 .SBTTL Create image procedure with parameters
00000008 OBE6 3224
OBE6 3225 ; define offsets for parameters
OBE6 3226
OBE6 3227 LNAM_DESC = 4
OBE6 3228 SLOT_TOP = 8
OBE6 3229 :++
OBE6 3230 : FUNCTIONAL DESCRIPTION:
OBE6 3231 :
OBE6 3232 : This routine is called when a executable image is to be run as a
OBE6 3233 : detached process. The parameters are passed by creating a mailbox
OBE6 3234 : with the same name as the process name (thereby making the name unique)
OBE6 3235 : and defining SYS$INPUT of the new image to be that mailbox.
OBE6 3236 : a detached process.
OBE6 3237 :
OBE6 3238 : CALLING SEQUENCE:
OBE6 3239 :
OBE6 3240 : PUSHAL PROC INFO SLOT
OBE6 3241 : PUSHAQ MBX_COGNAM_DESC
OBE6 3242 : CALLS #2,CRT_EXE_PARM
OBE6 3243 :
OBE6 3244 : INPUT PARAMETERS:
OBE6 3245 :
OBE6 3246 : LNAM_DESC(AP) - the address of the logical name to be associated with
OBE6 3247 : the mailbox holding the parameters
OBE6 3248 : SLOT_TOP(AP) - the address of the process info slot being used for
OBE6 3249 : this process
OBE6 3250 :
OBE6 3251 : IMPLICIT INPUTS:
OBE6 3252 :
OBE6 3253 : File specification information from GET_PARSE:
OBE6 3254 : FILE_SPEC_DESC - descriptor of the entire file specification
OBE6 3255 : PARM_DESC - descriptor of the parameters to be passed to the image.
OBE6 3256 :
OBE6 3257 : OUTPUT PARAMETERS:
OBE6 3258 :
OBE6 3259 : None.
OBE6 3260 :
OBE6 3261 : IMPLICIT OUTPUTS:
OBE6 3262 :
OBE6 3263 : None
OBE6 3264 :
OBE6 3265 : COMPLETION CODES:
OBE6 3266 :
OBE6 3267 : $$$_NORMAL
OBE6 3268 : Any completion code which may be produced by $CREPRC
OBE6 3269 :
OBE6 3270 : SIDE EFFECTS:
OBE6 3271 :
OBE6 3272 : A temporary mailbox will be created
OBE6 3273 : A detached process may be created
OBE6 3274 :--
OBE6 3275 :
OBE6 3276 CRT_EXE_PARM:
0004 OBE6 3277 .WORD *M<R2>
OBE8 3278
OBE8 3279 ; get the process info slot address
```

52	08	AC	D0	OBE8	3280	MOVL	SLOT_TOP(AP),R2	
				OBE8	3281			
				OBE8	3282			
				OBE8	3283			
				OBE8	3284			
045A'CF	00		D0	OBE8	3285	MOVL	#0,STSFLG	; clear the status flag
				OBF1	3286			
				OBF1	3287			
				OBF1	3288			
				OBF1	3289			
				OBF1	3290			
				OBF1	3291			
				OBF1	3292			
				OC0A	3293			
				OC0A	3294			
				OC0A	3295			
38	A2	01	C8	OC0A	3296	BISL2	#PRCINFO\$M_MBX_CHAN,PRCINFO\$M_FLAGS(R2)	; we're holding a mailbox ch
				OC0E	3297			
				OC0E	3298			
				OC0E	3299			
				OC0E	3300	\$QIOW_S	CHAN = PRCINFO\$M_MBX_CHAN(R2),-	; write to mailbox we just made
				OC0E	3301	FUNC	= #IOS\$WRITEVBLR!IOSM_NOW,-	; do the write now
				OC0E	3302	P1	= @<PARAM_DESC+DSC\$A_POINTER>,-	; the parameter string
				OC0E	3303	P2	= <PARAM_DESC+DSC\$W_LENGTH>	; size of parameter list
				OC30	3304			
				OC30	3305			
				OC30	3306			
0436'CF	04	AC	D0	OC30	3307	MOVL	LNAM_DESC(AP),INPUT	; input is the mailbox
				OC36	3308			
				OC36	3309			
				OC36	3310			
0432'CF	077D'CF		7E	OC36	3311	MOVAQ	FILE_SPEC_DESC,IMAGE	; run the actual image
				OC3D	3312			
				OC3D	3313			
				OC3D	3314			
				OC3D	3315	\$CREPRC_G	CREPRC_LIS	; create the process
				OC46	3316			
			04	OC46	3317	RET		; return with status in R0

```
0C47 3319 .SBTTL Create command procedure
0C47 3320
0C47 3321 ; define offsets for parameters
0C47 3322
0C47 3323 :++
0C47 3324 : FUNCTIONAL DESCRIPTION:
0C47 3325 :
0C47 3326 : This routine is called when a command procedure is to be run as a
0C47 3327 : detached process. The routine is started by VMS's LOGINOUT with
0C47 3328 : the filespec as the input file.
0C47 3329 :
0C47 3330 : CALLING SEQUENCE:
0C47 3331 :
0C47 3332 : CALLS #0,CRT_COM_PRC
0C47 3333 :
0C47 3334 : INPUT PARAMETERS:
0C47 3335 :
0C47 3336 : None
0C47 3337 :
0C47 3338 : IMPLICIT INPUTS:
0C47 3339 :
0C47 3340 : File specification information from GET_PARSE:
0C47 3341 : FILE_SPEC_DESC - descriptor of the entire file specification
0C47 3342 :
0C47 3343 : OUTPUT PARAMETERS:
0C47 3344 :
0C47 3345 : None.
0C47 3346 :
0C47 3347 : IMPLICIT OUTPUTS:
0C47 3348 :
0C47 3349 : None
0C47 3350 :
0C47 3351 : COMPLETION CODES:
0C47 3352 :
0C47 3353 : $$$_NORMAL
0C47 3354 : Any completion code which may be produced by $CREPRC
0C47 3355 :
0C47 3356 : SIDE EFFECTS:
0C47 3357 :
0C47 3358 : A detached process may be created
0C47 3359 :--
0C47 3360
0000 0C47 3361 CRT_COM_PRC:
0C47 3362 .WORD ^M<>
0C49 3363
0C49 3364 ; set up generic parameters for command file
0C49 3365
0432'CF 0473'CF 7E 0C49 3366 MOVAQ COM_IMAGE,IMAGE ; run under LOGINOUT
0C50 3367
0C50 3368 CLRL STSFLG ; re-init status flag
045A'CF 00000040 8F C8 0C54 3369 BISL2 #PRCSM_LOGIN,STSFLG ; bypass the UAF
0C5D 3370
0C5D 3371 ; put the name of the procedure in the parameter list
0C5D 3372
0436'CF 077D'CF 7E 0C5D 3373 MOVAQ FILE_SPEC_DESC,INPUT ; the procedure is the input
0C64 3374
0C64 3375 ; and create the process
```

TSTCNTRL
V04-000

TEST PACKAGE CONTROL PROGRAM
Create command procedure

D 7

16-SEP-1984 01:30:05
5-SEP-1984 04:26:03

VAX/VMS Macro V04-00
[UETP.SRC]UETPHAS00.MAR;1

Page 79
(39)

TS
V0

	0C64	3376			
	0C64	3377	\$CREPRC_G	CREPRC_LIS	; create the process
	0C6D	3378			
04	0C6D	3379	RET		; return with status in R0

```

00000004 OC6E 3381 .SBTTL Create command procedure with parameters
OC6E 3382
OC6E 3383 ; define offsets for parameters
OC6E 3384
OC6E 3385 TOT_PROC = 4
OC6E 3386
OC6E 3387 **
OC6E 3388 : FUNCTIONAL DESCRIPTION:
OC6E 3389 :
OC6E 3390 : This routine is called when a command procedure is to be run as a
OC6E 3391 : detached process, and that procedure has parameters with it. The
OC6E 3392 : routine passes the parameters by creating a one line command
OC6E 3393 : procedure (which has a unique one filename) which simply starts up
OC6E 3394 : the original command procedure along with the parameters.
OC6E 3395 :
OC6E 3396 : CALLING SEQUENCE:
OC6E 3397 :
OC6E 3398 : PUSHAL TOT_PROC RUN
OC6E 3399 : CALLS #1,CRT_COM_PARM
OC6E 3400 :
OC6E 3401 : INPUT PARAMETERS:
OC6E 3402 :
OC6E 3403 : TOT_PROC(AP) - address of the cumulative number of total processes
OC6E 3404 : run this session.
OC6E 3405 :
OC6E 3406 : IMPLICIT INPUTS:
OC6E 3407 :
OC6E 3408 : File specification information from GET_PARSE. Including --
OC6E 3409 : FILE_SPEC_DESC - descriptor of the entire file specification
OC6E 3410 : FILENAME_DESC - descriptor of the actual filename
OC6E 3411 : PARAM_DESC - descriptor of the parameter string
OC6E 3412 :
OC6E 3413 : OUTPUT PARAMETERS:
OC6E 3414 :
OC6E 3415 : None.
OC6E 3416 :
OC6E 3417 : IMPLICIT OUTPUTS:
OC6E 3418 :
OC6E 3419 : None.
OC6E 3420 :
OC6E 3421 : COMPLETION CODES:
OC6E 3422 :
OC6E 3423 : $$$_NORMAL
OC6E 3424 : any completion which may be produced by $CREPROC
OC6E 3425 :
OC6E 3426 : SIDE EFFECTS:
OC6E 3427 :
OC6E 3428 : Makes a temporary command file (which is to be deleted by PROC_TERM).
OC6E 3429 : May create a detached process.
OC6E 3430 :
OC6E 3431 : --
OC6E 3432 : CRT_COM_PARM:
OC6E 3433 : .WORD ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
OC70 3434
OC70 3435 ; create a detached process for command file with parameters
OC70 3436
OC70 3437 ; initialize the stack area for local storage
OC70 3438
OC70 3439
OC70 3440
OC70 3441
OC70 3442
OC70 3443
OC70 3444
OC70 3445
OC70 3446
OC70 3447
OC70 3448
OC70 3449
OC70 3450
OC70 3451
OC70 3452
OC70 3453
OC70 3454
OC70 3455
OC70 3456
OC70 3457
OC70 3458
OC70 3459
OC70 3460
OC70 3461
OC70 3462
OC70 3463
OC70 3464
OC70 3465
OC70 3466
OC70 3467
OC70 3468
OC70 3469
OC70 3470
OC70 3471
OC70 3472
OC70 3473
OC70 3474
OC70 3475
OC70 3476
OC70 3477
OC70 3478
OC70 3479
OC70 3480
OC70 3481
OC70 3482
OC70 3483
OC70 3484
OC70 3485
OC70 3486
OC70 3487
OC70 3488
OC70 3489
OC70 3490
OC70 3491
OC70 3492
OC70 3493
OC70 3494
OC70 3495
OC70 3496
OC70 3497
OC70 3498
OC70 3499
OC70 3500
OC70 3501
OC70 3502
OC70 3503
OC70 3504
OC70 3505
OC70 3506
OC70 3507
OC70 3508
OC70 3509
OC70 3510
OC70 3511
OC70 3512
OC70 3513
OC70 3514
OC70 3515
OC70 3516
OC70 3517
OC70 3518
OC70 3519
OC70 3520
OC70 3521
OC70 3522
OC70 3523
OC70 3524
OC70 3525
OC70 3526
OC70 3527
OC70 3528
OC70 3529
OC70 3530
OC70 3531
OC70 3532
OC70 3533
OC70 3534
OC70 3535
OC70 3536
OC70 3537
OC70 3538
OC70 3539
OC70 3540
OC70 3541
OC70 3542
OC70 3543
OC70 3544
OC70 3545
OC70 3546
OC70 3547
OC70 3548
OC70 3549
OC70 3550
OC70 3551
OC70 3552
OC70 3553
OC70 3554
OC70 3555
OC70 3556
OC70 3557
OC70 3558
OC70 3559
OC70 3560
OC70 3561
OC70 3562
OC70 3563
OC70 3564
OC70 3565
OC70 3566
OC70 3567
OC70 3568
OC70 3569
OC70 3570
OC70 3571
OC70 3572
OC70 3573
OC70 3574
OC70 3575
OC70 3576
OC70 3577
OC70 3578
OC70 3579
OC70 3580
OC70 3581
OC70 3582
OC70 3583
OC70 3584
OC70 3585
OC70 3586
OC70 3587
OC70 3588
OC70 3589
OC70 3590
OC70 3591
OC70 3592
OC70 3593
OC70 3594
OC70 3595
OC70 3596
OC70 3597
OC70 3598
OC70 3599
OC70 3600
OC70 3601
OC70 3602
OC70 3603
OC70 3604
OC70 3605
OC70 3606
OC70 3607
OC70 3608
OC70 3609
OC70 3610
OC70 3611
OC70 3612
OC70 3613
OC70 3614
OC70 3615
OC70 3616
OC70 3617
OC70 3618
OC70 3619
OC70 3620
OC70 3621
OC70 3622
OC70 3623
OC70 3624
OC70 3625
OC70 3626
OC70 3627
OC70 3628
OC70 3629
OC70 3630
OC70 3631
OC70 3632
OC70 3633
OC70 3634
OC70 3635
OC70 3636
OC70 3637
OC70 3638
OC70 3639
OC70 3640
OC70 3641
OC70 3642
OC70 3643
OC70 3644
OC70 3645
OC70 3646
OC70 3647
OC70 3648
OC70 3649
OC70 3650
OC70 3651
OC70 3652
OC70 3653
OC70 3654
OC70 3655
OC70 3656
OC70 3657
OC70 3658
OC70 3659
OC70 3660
OC70 3661
OC70 3662
OC70 3663
OC70 3664
OC70 3665
OC70 3666
OC70 3667
OC70 3668
OC70 3669
OC70 3670
OC70 3671
OC70 3672
OC70 3673
OC70 3674
OC70 3675
OC70 3676
OC70 3677
OC70 3678
OC70 3679
OC70 3680
OC70 3681
OC70 3682
OC70 3683
OC70 3684
OC70 3685
OC70 3686
OC70 3687
OC70 3688
OC70 3689
OC70 3690
OC70 3691
OC70 3692
OC70 3693
OC70 3694
OC70 3695
OC70 3696
OC70 3697
OC70 3698
OC70 3699
OC70 3700
OC70 3701
OC70 3702
OC70 3703
OC70 3704
OC70 3705
OC70 3706
OC70 3707
OC70 3708
OC70 3709
OC70 3710
OC70 3711
OC70 3712
OC70 3713
OC70 3714
OC70 3715
OC70 3716
OC70 3717
OC70 3718
OC70 3719
OC70 3720
OC70 3721
OC70 3722
OC70 3723
OC70 3724
OC70 3725
OC70 3726
OC70 3727
OC70 3728
OC70 3729
OC70 3730
OC70 3731
OC70 3732
OC70 3733
OC70 3734
OC70 3735
OC70 3736
OC70 3737
OC70 3738
OC70 3739
OC70 3740
OC70 3741
OC70 3742
OC70 3743
OC70 3744
OC70 3745
OC70 3746
OC70 3747
OC70 3748
OC70 3749
OC70 3750
OC70 3751
OC70 3752
OC70 3753
OC70 3754
OC70 3755
OC70 3756
OC70 3757
OC70 3758
OC70 3759
OC70 3760
OC70 3761
OC70 3762
OC70 3763
OC70 3764
OC70 3765
OC70 3766
OC70 3767
OC70 3768
OC70 3769
OC70 3770
OC70 3771
OC70 3772
OC70 3773
OC70 3774
OC70 3775
OC70 3776
OC70 3777
OC70 3778
OC70 3779
OC70 3780
OC70 3781
OC70 3782
OC70 3783
OC70 3784
OC70 3785
OC70 3786
OC70 3787
OC70 3788
OC70 3789
OC70 3790
OC70 3791
OC70 3792
OC70 3793
OC70 3794
OC70 3795
OC70 3796
OC70 3797
OC70 3798
OC70 3
```

6E	011D 8F	00	00 8F	00	2C	0C77	3438		
			5B	5E	D0	0C77	3439	MOVCS #0,#0,#0,#COM_PARM_LENGTH,(SP) ; clear out local storage	
						0C80	3440	MOVL SP,R11 ; keep pointer to local storage	
						0C83	3441		
						0C83	3442	; first set up generic parameters for command file	
	0432'CF	0473'CF			7E	0C83	3443		
						0C83	3444	MOVAQ COM_IMAGE,IMAGE ; run under LOGINOUT	
						0C8A	3445		
	045A'CF	045A'CF			D4	0C8A	3446	CLRL STSFLG ; re-init status flag	
	00000040	8F			C8	0C8E	3447	BISL2 #PRCSM_LOGIN,STSFLG ; bypass the UAF	
						0C97	3448		
						0C97	3449	; since we must pass parameters, we will make a command file which	
						0C97	3450	; calls the command file we want to start and passes the parameters	
						0C97	3451	; along to it. It will also capture the exit status of the command	
						0C97	3452	; file we want to start and will pass that status back to the TSTCNTRL.	
						0C97	3453	; The file that we make will look like:	
						0C97	3454	;	
						0C97	3455	;\$@filespec -	
						0C97	3456	;\$parameter_list	
						0C97	3457	;\$ status = \$status	
						0C97	3458	;\$ exit status	
						0C97	3459	; initialize descriptor for the temporary command filename	
						0C97	3460		
	08 AB				DE	0C97	3461	MOVAL TEMP_COM_STR(R11),- ; put the address in the	
	04 AB					0C9A	3462	<TEMP_COM_DESC+DS(\$A_POINTER)>(R11) ; pointer area	
						0C9C	3463		
						0C9C	3464	; make the filename for the temporary command file	
						0C9C	3465		
	089C'CF				7F	0C9C	3466	PUSHAQ FILENAME_DESC ; the test filename	
	04 AC				DD	0CA0	3467	PUSHL TOT_PROCTAP ; the ID number	
	0467'CF				7F	0CA3	3468	PUSHAQ TMP_COM_EXT_DESC ; the file extension	
					7F	0CA7	3469	PUSHAQ TEMP_COM_DESC(R11) ; where the new filename goes	
	0E35'CF	04			FB	0CA9	3470	CALLS #4,MAKE_TMP_LOG_NAM ; and make the name	
						0CAE	3471		
						0CAE	3472	; set up the record to handle .COM files with parameters by using	
						0CAE	3473	; COMMENT_DESC, by putting a \$@ at the beginning of the filespec	
						0CAE	3474	; and concatenating a hyphen and a blank to the end of file spec	
						0CAE	3475		
						0CAE	3476	; set up an index to use to point to various pieces in the new record	
						0CAE	3477		
					57	D4	0CAE	3478	CLRL R7 ; index into new record
						0CB0	3479		
						0CB0	3480	; then initialize the descriptor for the indirect file spec	
						0CB0	3481		
						0CB0	3482	MOVAL COM_LINE_STR(R11),- ; put the address in the	
						0CB3	3483	<COM_LINE_DESC+DS(\$A_POINTER)>(R11) ; pointer area	
						0CB5	3484		
						0CB5	3485	; now, move '\$@' into the string	
						0CB5	3486		
	1D AB47	24			90	0CB5	3487	MOVB #^A/\$/,COM_LINE_STR(R11)[R7] ; put \$ in front	
		57			B6	0CBA	3488	INCW R7 ; update the index	
						0CBC	3489		
	1D AB47	40	8F		90	0CBC	3490	MOVB #^A/@/,COM_LINE_STR(R11)[R7] ; put @ in front of spec	
		57	B6			0CC2	3491	INCW R7 ; update the index	
						0CC4	3492		
						0CC4	3493		
						0CC4	3494	; now, put the complete filespec in	

077D'CF	28	OCC4	3495	MOV C3	<FILE_SPEC_DESC+DSC\$W_LENGTH>,-	; the size of the filespec
0781'DF		OCC8	3496		@<FILE_SPEC_DESC+DSC\$A_POINTER>,-	; take it from here
1D AB47		OCCB	3497		COM_LINE_STR(R11)[R7]	; and put it here
		OCCE	3498			
57 077D'CF	A0	OCCE	3499	ADDW2	<FILE_SPEC_DESC+DSC\$W_LENGTH>,R7	; update the index
		OCD3	3500			
		OCD3	3501			
		OCD3	3502			; and finally put the " -" on the end
1D AB47	20	90	OCD3	MOV B	#^A/ /,COM_LINE_STR(R11)[R7]	; add blank to the end
	57	B6	OCD8	INCW	R7	; update the index
			OCDA			
1D AB47	20	90	OCDA	MOV B	#^A/-/,COM_LINE_STR(R11)[R7]	; add continuation character
	57	B6	OCDF	INCW	R7	; update the index (and the size)
			OCE1			
15 AB	57	3C	OCE1	MOVZWL	R7,<COM_LINE_DESC+DSC\$W_LENGTH>(R11)	; put the size in the desc
			OCE5			
			OCE5			; now create the temporary command file, and write the proper
			OCE5			; records into it.
			OCE5			
			OCE5	\$FAB_STORE	FAB = COM_FAB,-	; point to temporary command file
			OCE5		FNA = TEMP_COM_STR(R11),-	; the filename
			OCE5		FNS = TEMP_COM_DESC(R11)	; the size
			OCF3			
			OCF3	\$CREATE	FAB = COM_FAB,-	; now make the file
			OCF3		ERR = RMSERR	
			OD02			
			OD02	\$CONNECT	RAB = COM_RAB,-	; connect the stream
			OD02		ERR = RMSERR	
			OD11			
			OD11	\$RAB_STORE	RAB = COM_RAB,-	; point to the filespec which we
			OD11		RSZ = COM_LINE_DESC(R11),-	; really want to use
			OD11		RBF = COM_LINE_STR(R11)	
			OD20			
			OD20	\$PUT	RAB = COM_RAB,-	; write that record
			OD20		ERR = RMSERR	
			OD2F			
			OD2F	\$RAB_STORE	RAB = COM_RAB,-	; point to the parameter list
			OD2F		RSZ = PARAM_DESC,-	; associated with the above
			OD2F		RBF = @<PARAM_DESC+DSC\$A_POINTER>	; command procedure
			OD40			
			OD40	\$PUT	RAB = COM_RAB,-	; write that record
			OD40		ERR = RMSERR	
			OD4F			
			OD4F	\$RAB_STORE	RAB = COM_RAB,-	; point to the record holding
			OD4F		RSZ = COM_STAT_DESC,-	; DCL to save the final status of
			OD4F		RBF = @<COM_STAT_DESC+DSC\$A_POINTER>	; the command procedure
			OD60			
			OD60	\$PUT	RAB = COM_RAB,-	; put that record
			OD60		ERR = RMSERR	
			OD6F			
			OD6F	\$RAB_STORE	RAB = COM_RAB,-	; point to the record holding
			OD6F		RSZ = COM_EXIT_DESC,-	; the exit from the command
			OD6F		RBF = @<COM_EXIT_DESC+DSC\$A_POINTER>	; procedure
			OD80			
			OD80	\$PUT	RAB = COM_RAB,-	; put that record
			OD80		ERR = RMSERR	
			OD8F			

```
0436'CF 6B 7E 0D8F 3552 $CLOSE FAB = COM_FAB,- ; close the file
0D8F 3553 ERR = RMSERR
0D9E 3554
0D9E 3555 ; we now have our indirect procedure to start up the want command
0D9E 3556 ; procedure and pass parameters to it
0D9E 3557
0D9E 3558 ; now let the input of LOGINOUT be the file we just made
0D9E 3559
0D9E 3560 MOVAQ TEMP_COM_DESC(R11),INPUT ; input is temp command file
0DA3 3561
0DA3 3562 ; and finally start up the procedure
0DA3 3563
0DA3 3564 $CREPRC_G CREPRC_LIS ; . . . and create the process
0DAC 3565
55 50 D0 0DAC 3566 MOVL R0,R5 ; save the completion status
0DAF 3567
0DAF 3568 ; see if create process worked. If it didn't, then delete the
0DAF 3569 ; temporary command procedure; otherwise, the process termination
0DAF 3570 ; routine will delete it.
0DAF 3571
55 01 D1 0DAF 3572 CMPL #SS$ NORMAL,R5 ; did the process get created
OF 13 0DB2 3573 BEQL END_CRT_COM_PARM ; yes, so leave
0DB4 3574
0DB4 3575 $ERASE FAB = COM_FAB,- ; no, so delete the temporary
0DB4 3576 ERR = RMSERR ; command procedure
0DC3 3577
0DC3 3578 END_CRT_COM_PARM:
0DC3 3579
50 55 D0 0DC3 3580 MOVL R5,R0 ; replace the status code
0DC6 3581
04 0DC6 3582 RET ; return with status in R0
0DC7 3583
```

```
ODC7 3585      .SBTTL Create Process-name
ODC7 3586
ODC7 3587      ; define offsets for parameters
ODC7 3588
00000004 ODC7 3589      CP_PROCESS_NAME = 4
00000008 ODC7 3590      CP_ID_NUMBER   = 8
0000000C ODC7 3591      CP_TEST_FILENAME = 12
ODC7 3592
ODC7 3593      :++
ODC7 3594      : FUNCTIONAL DESCRIPTION:
ODC7 3595      :
ODC7 3596      : This subroutine is called to create the unique process name before the
ODC7 3597      : creation of a detached process. It does this by taking the filename
ODC7 3598      : and concatenating an ID number which is generated by the TSTCNTRL.
ODC7 3599      : Though the filenames may be repeated, the ID number is unique.
ODC7 3600
ODC7 3601      : CALLING SEQUENCE:
ODC7 3602      :
ODC7 3603      : This routine is called by CREAT_PROC via:
ODC7 3604      :     PUSHAL TEST_FILENAME
ODC7 3605      :     PUSHAL ID_NUMBER
ODC7 3606      :     PUSHAL PROCESS_NAME
ODC7 3607      :     CALLS #3,MAKE_PROC_NAM
ODC7 3608
ODC7 3609      : INPUT PARAMETERS:
ODC7 3610      :
ODC7 3611      :     CP_PROCESS_NAME(AP) - address of descriptor to hold process name
ODC7 3612      :     CP_ID_NUMBER(AP)   - address of longword holding ID number
ODC7 3613      :     CP_TEST_FILENAME(AP) - address of descriptor holding filename (without exten
ODC7 3614
ODC7 3615      : IMPLICIT INPUTS:
ODC7 3616      :
ODC7 3617      :     None.
ODC7 3618
ODC7 3619      : OUTPUT PARAMETERS:
ODC7 3620      :
ODC7 3621      :     The process name (@CP_PROCESS_NAME(AP)).
ODC7 3622      :     Completion code in R0.
ODC7 3623
ODC7 3624      : IMPLICIT OUTPUTS:
ODC7 3625      :
ODC7 3626      :     None.
ODC7 3627
ODC7 3628      : COMPLETION CODES:
ODC7 3629      :
ODC7 3630      :     SSS_NORMAL, OTSS_OUTCONERR
ODC7 3631
ODC7 3632      : SIDE EFFECTS:
ODC7 3633      :
ODC7 3634      :     None.
ODC7 3635      :--
ODC7 3636
0FFC ODC7 3637 MAKE_PROC_NAM:
ODC7 3638      .WORD ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
ODC9 3639
ODC9 3640      ; first pad with leading blanks (if needed)
ODC9 3641
```

SA	OC	AC	D0	ODC9	3642	MOVL	CP TEST_FILENAME(AP),R10	; get addr of desc. for filename			
58	6A	9A	ODCD	3643	MOVZBL	(R10),R8	; get size of filnam (clear desc typ				
09	58	D1	ODD0	3644	CMPL	R8,#PNAME_FILNAM_SIZ	; is the size of filnam to large?				
	03	15	ODD3	3645	BLEQ	10\$	; no, so continue				
58	09	D0	ODD5	3646	MOVL	#PNAME_FILNAM_SIZ,R8	; yes, so make filnam max allowed				
			ODD8	3647							
5B	04	AC	D0	ODD8	3648	10\$: MOVL	CP_PROCESS_NAME(AP),R11	; get addr of desc for procname			
				ODDC	3649						
	56	D4	ODDC	3650	CLRL	R6	; this is our counter and index				
54	09	58	C3	ODDE	3651	SUBL3	R8,#PNAME_FILNAM_SIZ,R4	; number of blanks needed			
		09	13	ODE2	3652	BEQL	30\$	; no blanks are needed			
				ODE4	3653						
04	BB46	20	90	ODE4	3654	20\$: MOVB	#^A/ /,@DSC\$A_POINTER(R11)[R6]	; put blanks in			
F7	56	54	F2	ODE9	3655	AOBLSS	R4,R6,20\$	; are all of them in?			
				ODED	3656						
				ODED	3657			; now move the filename in, truncate it if too large.			
				ODED	3658						
04	BA	58	28	ODED	3659	30\$: MOVCL	R8,@DSC\$A_POINTER(R10),-				
04	BB46			ODF1	3660		@DSC\$A_POINTER(R11)[R6]	; move the filename in			
56	58	C0	ODF4	3661	ADDL2	R8,R6		; move index to end of string			
				ODF7	3662						
				ODF7	3663			; add the underscore			
				ODF7	3664						
04	BB46	5F	8F	90	ODF7	3665	MOVB	#^A/_ /,@DSC\$A_POINTER(R11)[R6]	; put underscore in		
		56	D6	ODFD	3666	INCL	R6	; point to next character			
				ODFF	3667						
				ODFF	3668			; now put the ID number in			
				ODFF	3669						
6E	10	00	00	5E	10	C2	ODFF	3670	SUBL2	#ID_DESC_LENGTH,SP	; allocate space for local storage
		8F	00	2C	OE02	3671	MOVCL	#0,#0,#0,#ID_DESC_LENGTH,(SP)	; clear out local storage		
		57	5E	D0	OE09	3672	MOVL	SP,R7	; keep the pointer to this area		
	04	A7	08	A7	DE	OE0C	3673	MOVAL	ID_STR(R7),ID_ADDR(R7)	; initialize the addr of desc.	
		67	04	D0	OE11	3674	MOVL	#ID_STR_SIZ,ID_DESC(R7)	; initialize size of desc.		
					OE14	3675					
	0C	A7	08	BC	3C	OE14	3676	MOVZWL	@CP_ID_NUMBER(AP),ID_NUMB(R7)	; clear upper word of ID number	
					OE19	3677					
		04		DD	OE19	3678	PUSHL	#ID_STR_SIZ	; this number of ascii digits		
		67		DF	OE1B	3679	PUSHAL	ID_DESC(R7)	; where the converted string goes		
		0C	A7	DF	OE1D	3680	PUSHAL	ID_NUMB(R7)	; the number to convert		
00000000'GF		03	FB	OE20	3681	CALLS	#3,G^OTSS\$CVT_L_TI	; convert the number to text			
				OE27	3682						
				OE27	3683			; now move ID into process name			
				OE27	3684						
	08	A7	04	28	OE27	3685	MOVCL	#ID_STR_SIZ,ID_STR(R7),-			
		04	BB46		OE2B	3686		@DSC\$A_POINTER(R11)[R6]	; move ID in		
					OE2E	3687					
		0E		D0	OE2E	3688	MOVL	#<PROCESS NAME_SIZ-1>,-			
		6B			OE30	3689		DSC\$W_LENGTH(R11)	; put size of process name in		
					OE31	3690			; subtract one so a blank looks		
					OE31	3691			; like it is there.		
					OE31	3692					
	50	01	D0	OE31	3693	MOVL	#SS\$_NORMAL,R0	; signify success			
					OE34	3694					
			04	OE34	3695	RET					

```
00000004 OE35 3697 .SBTTL Make temporary file name
00000008 OE35 3698 ; define offsets for parameters
0000000C OE35 3699
00000010 OE35 3700 MF_TEMP_FILE = 4
OE35 3701 MF_TEMP_EXT = 8
OE35 3702 MF_ID_NUMBER = 12
OE35 3703 MF_TEST_FILENAME = 16
OE35 3704
OE35 3705 ;++
OE35 3706 : FUNCTIONAL DESCRIPTION:
OE35 3707 :
OE35 3708 : This routine is called to make the filespec for the temporary
OE35 3709 : files produced by the TSTCNTRL. It makes each filespec
OE35 3710 : unique by appending the sequence number to the spec.
OE35 3711 :
OE35 3712 : CALLING SEQUENCE:
OE35 3713 :
OE35 3714 : This routine is called by CREAT_PROC via:
OE35 3715 :     PUSHAQ TEST_FILENAME
OE35 3716 :     PUSHAL ID_NUMBER
OE35 3717 :     PUSHAQ TEMP_EXT
OE35 3718 :     PUSHAQ TEMP_FILE
OE35 3719 :     CALLS #4,MAKE_TMP_LOG_NAM
OE35 3720 :
OE35 3721 : INPUT PARAMETERS:
OE35 3722 :
OE35 3723 :     MF_TEMP_FILE(AP) - address of descriptor to hold temporary log filename
OE35 3724 :     MF_TEMP_EXT(AP) - address of descriptor holding the temp file extension
OE35 3725 :     MF_ID_NUMBER(AP) - address of longword holding ID number
OE35 3726 :     MF_TEST_FILENAME(AP) - address of descriptor holding filename (without exten
OE35 3727 :
OE35 3728 : IMPLICIT INPUTS:
OE35 3729 :
OE35 3730 :     None.
OE35 3731 :
OE35 3732 : OUTPUT PARAMETERS:
OE35 3733 :
OE35 3734 :     The temporary filename (@MF_TEMP_FILE(AP)).
OE35 3735 :     Completion code in R0.
OE35 3736 :
OE35 3737 : IMPLICIT OUTPUTS:
OE35 3738 :
OE35 3739 :     None.
OE35 3740 :
OE35 3741 : COMPLETION CODES:
OE35 3742 :
OE35 3743 :     SSS_NORMAL, OTSS_OUTCONERR
OE35 3744 :
OE35 3745 : SIDE EFFECTS:
OE35 3746 :
OE35 3747 :     None.
OE35 3748 : --
OE35 3749 :
OE35 3750 MAKE_TMP_LOG_NAM:
OFFC OE35 3751 .WORD ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
OE37 3752
OE37 3753 ; get input information
```

```

59 08 AC D0 0E37 3754
5A 10 AC D0 0E37 3755
58 58 6A 9A 0E3B 3756
5B 04 AC D0 0E3F 3757
                                0E42 3758
                                0E46 3759
                                0E46 3760
                                0E46 3761
05 58 D1 0E46 3762
                                0E49 3763
                                0E49 3764
58 05 D0 0E4B 3765
                                0E4E 3766
                                0E4E 3767 10$:
04 BA 58 D0 0E4E 3768
56 56 D4 0E51 3769
04 BA 58 28 0E53 3770
04 BB 46 0E57 3771
56 58 C0 0E5A 3772
                                0E5D 3773
                                0E5D 3774
                                0E5D 3775
6E 10 00 00 5E 10 C2 0E5D 3776
8F 00 2C 0E60 3777
57 5E D0 0E67 3778
04 A7 08 A7 DE 0E6A 3779
67 04 D0 0E6F 3780
                                0E72 3781
0C A7 0C BC 3C 0E72 3782
                                0E77 3783
                                04 DD 0E77 3784
67 DF 0E79 3785
0C A7 DF 0E7B 3786
00000000'GF 03 FB 0E7E 3787
                                0E85 3788
                                0E85 3789
                                0E85 3790
08 A7 C4 28 0E85 3791
04 BB 46 0E89 3792
56 04 C0 0E8C 3793
68 04 C0 0E8F 3794
                                0E92 3795
                                0E92 3796
                                0E92 3797
04 89 69 28 0E92 3798
04 BB 46 0E96 3799
68 69 A0 0E99 3800
                                0E9C 3801
50 01 D0 0E9C 3802
                                0E9F 3803
                                04 0E9F 3804
                                0EA0 3805

                                MOVL MF_TEMP_EXT(AP),R9 ; get addr of desc of extension
                                MOVL MF_TEST_FILENAME(AP),R10 ; get addr of desc. for filename
                                MOVZBL DSC$W_LENGTH(R10),R8 ; get size of filnam (clear desc typ
                                MOVL MF_TEMP_FILE(AP),R11 ; get addr of desc for log filnam

                                ; move up to MAX_TMP_LOG_NAM characters into file spec
                                CMPL R8,#MAX_TMP_LOG_NAM ; is size of filename more than
                                ; we can handle?
                                BLEQ 10$ ; no, so don't truncate
                                MOVL #MAX_TMP_LOG_NAM,R8 ; truncate it down to acceptable siz

                                MOVL R8,DSC$W_LENGTH(R11) ; keep size of growing filespec
                                CLRL R6 ; this is our index and pointer
                                MOVCL3 R8,@DSC$A_POINTER(R10),-
                                @DSC$A_POINTER(R11)[R6] ; move filename in
                                ADDL2 R8,R6 ; adjust pointer

                                ; now get the ID number
                                SUBL2 #ID_DESC_LENGTH,SP ; allocate space for local storage
                                MOVCL5 #0,#0,#0,#ID_DESC_LENGTH,(SP) ; clear out local storage
                                MOVL SP,R7 ; keep the pointer to this area
                                MOVAL ID_STR(R7),ID_ADDR(R7) ; initialize the addr of desc.
                                MOVL #ID_STR_SIZE,ID_DESC(R7) ; initialize size of desc.

                                10VZWL @MF_ID_NUMBER(AP),ID_NUMB(R7) ; clear out upper word of ID

                                PUSHCL #ID_STR_SIZE ; this number of ascii digits
                                PUSHAL ID_DESC(R7) ; where the converted string goes
                                PUSHAL ID_NUMB(R7) ; the number to convert
                                CALLS #3,G^OTSS$CVT_L_TI ; convert the number to text

                                ; now move ID into process name
                                MOVCL3 #ID_STR_SIZE,ID_STR(R7),-
                                @DSC$A_POINTER(R11)[R6] ; move ID in
                                ADDL2 #ID_STR_SIZE,R6 ; adjust pointer
                                ADDL2 #ID_STR_SIZE,DSC$W_LENGTH(R11) ; adjust size

                                ; now put file extension on
                                MOVCL3 DSC$W_LENGTH(R9),@DSC$A_POINTER(R9),-
                                @DSC$A_POINTER(R11)[R6] ; put extension on
                                ADDW2 DSC$W_LENGTH(R9),DSC$W_LENGTH(R11) ; adjust size

                                MOVL #SS$_NORMAL,R0 ; return success
                                RET
```

```
00000004 OEA0 3807 .SBTTL Assignment Expression Subroutine
          OEA0 3808
          OEA0 3809 ; define offsets for parameters
          OEA0 3810
          OEA0 3811 ASG_CNT_CAS = 4
          OEA0 3812 ;++
          OEA0 3813 : FUNCTIONAL DESCRIPTION:
          OEA0 3814 :
          OEA0 3815 : This routine is reached when the CASE$V_ASSIGN bit is set. It
          OEA0 3816 : takes the results of the parse routine (which has already determined
          OEA0 3817 : the keyword and if it valid or not) and calls the proper assignment
          OEA0 3818 : routine for that particular keyword.
          OEA0 3819 :
          OEA0 3820 : CALLING SEQUENCE:
          OEA0 3821 :
          OEA0 3822 : PUSHAL CONTROL_CASE
          OEA0 3823 : CALLS #1,ASSIGN_SBRTN
          OEA0 3824 :
          OEA0 3825 : INPUT PARAMETERS:
          OEA0 3826 :
          OEA0 3827 : ASG_CNT_CAS(AP) - address of the longword holding the CONTROL_CASE
          OEA0 3828 :
          OEA0 3829 : IMPLICIT INPUTS:
          OEA0 3830 :
          OEA0 3831 : The results of the parse routine are implicit inputs, namely
          OEA0 3832 : KEYWRD_EQ_VAL - a word which holds a numerical value equivalent
          OEA0 3833 : to each keyword. These values are defined in the GET_PARSE
          OEA0 3834 : routine.
          OEA0 3835 : KEYVALUE_DESC - a descriptor which points to the value of the
          OEA0 3836 : assignment statement
          OEA0 3837 :
          OEA0 3838 : OUTPUT PARAMETERS:
          OEA0 3839 :
          OEA0 3840 : None
          OEA0 3841 :
          OEA0 3842 : IMPLICIT OUTPUTS:
          OEA0 3843 :
          OEA0 3844 : This routine itself has no outputs; however, the routines which this
          OEA0 3845 : one calls may modify global variables. These modifications are
          OEA0 3846 : described in the subroutines.
          OEA0 3847 :
          OEA0 3848 : COMPLETION CODES:
          OEA0 3849 :
          OEA0 3850 : SSS_NORMAL
          OEA0 3851 :
          OEA0 3852 : SIDE EFFECTS:
          OEA0 3853 :
          OEA0 3854 : None
          OEA0 3855 : --
          OEA0 3856 :
          0004 OEA0 3857 ASSIGN_SBRTN:
          OEA0 3858 .WORD ^M<R2>
          OEA2 3859
          OEA2 3860 ; turn off assignment flag and turn on parse flag
          OEA2 3861
          OEA2 3862 BICL2 #CASE$M_ASSIGN,@ASG_CNT_CAS(AP) ; don't come back to assign
          04 BC 00000080 8F C8 OEA6 3863 BISL2 #CASE$M_PARSE,@ASG_CNT_CAS(AP) ; but go to parse
```

```

      OEAE 3864
      OEAE 3865      ; now by using the results of the parse routine
      OEAE 3866      ; case to proper assignment
      OEAE 3867
00    EA  OEAE 3868      FFS    #0,-      ; start looking at position 0
20    OEBO 3869      #<ASSIGN$$ ASGN_CASE*8>,-      ; this is how many bits
04A7'CF OEBO 3870      ASSIGN_CASE,-      ; this is the base
52    OEBO 3871      R2      ; put results here
06    00 52 8F OEBO 3872
      OEBO 3873      CASEB   R2,#0,#<ASSIGN$K_ASGN_CASE_SIZ - 1> ; goto proper routine
      OEBO 3874 10$:
0011' OEBO 3875      .WORD   LOG_ASG    - 10$
001D' OEBO 3876      .WORD   PNAME_ASG  - 10$
0029' OEBO 3877      .WORD   START_ASG  - 10$
0035' OEBO 3878      .WORD   STOP_ASG   - 10$
0041' OEBO 3879      .WORD   COM_ASG    - 10$
004D' OEBO 3880      .WORD   TIM_ASG    - 10$
0059' OEBO 3881      .WORD   PAR_ASG    - 10$
      OEBO 3882
0054 31 OEBO 3883      BRW     END_ASSIGN_SBRTN      ; if no match, just exit
      OEBO 3884
      OEBO 3885 LOG_ASG:
      OEBO 3886      ; do assignment to change name of the permanent log file
      OEBO 3887
077D'CF 7F OEBO 3888      PUSHAQ  FILE_SPEC_DESC      ; location of new filename
0F98'CF 01 FB OEBO 3889      CALLS   #1,LOG_ASG_RTN      ; do the assignment
      OEBO 3890
0048 31 OEBO 3891      BRW     END_ASSIGN_SBRTN      ; and leave
      OEBO 3892
      OEBO 3893 PNAME_ASG:
      OEBO 3894      ; do assignment for the process name
      OEBO 3895
08B4'CF 7F OEBO 3896      PUSHAQ  ASSIGN_VALUE_DESC      ; the location of the new name
0F22'CF 01 FB OEBO 3897      CALLS   #1,PNAME_ASG_RTN      ; do the assignment
      OEBO 3898
003C 31 OEBO 3899      BRW     END_ASSIGN_SBRTN      ; and leave
      OEBO 3900
      OEBO 3901 START_ASG:
      OEBO 3902      ; do assignment to change the starting sentinel
      OEBO 3903
08B4'CF 7F OEBO 3904      PUSHAQ  ASSIGN_VALUE_DESC      ; the location of the new sentinel
0F43'CF 01 FB OEBO 3905      CALLS   #1,START_ASG_RTN      ; do the assignment
      OEBO 3906
0030 31 OEBO 3907      BRW     END_ASSIGN_SBRTN      ; and leave
      OEBO 3908
      OEBO 3909 STOP_ASG:
      OEBO 3910      ; do assignment to change the ending sentinel
      OEBO 3911
08B4'CF 7F OEBO 3912      PUSHAQ  ASSIGN_VALUE_DESC      ; the location of the new sentinel
0F59'CF 01 FB OEBO 3913      CALLS   #1,STOP_ASG_RTN      ; do the assignment
      OEBO 3914
0024 31 OEBO 3915      BRW     END_ASSIGN_SBRTN      ; and leave
      OEBO 3916
      OEBO 3917 COM_ASG:
      OEBO 3918      ; do assignment to determine if we turn on or off the comment flag
      OEBO 3919
08B4'CF 7F OEBO 3920      PUSHAQ  ASSIGN_VALUE_DESC      ; the location of the flag
```

TEST PACKAGE CONTROL PROGRAM
Assignment Expression Subroutine

16-SEP-1984 01:30:05
5-SEP-1984 04:26:03

VAX/VMS Macro V04-00
[UETP.SRC]UETPHAS00.MAR:1

Page 90
(43)

TS
VO

```

0F6F'CF    01   FB   OEFE   3921      CALLS    #1,COM_ASG_RTN          ; do the assignment
                                OF03   3922
                                0018   31   OF03   3923      BRW      END_ASSIGN_SBR TN        ; and leave
                                OF06   3924
                                OF06   3925 TIM_ASG:
                                OF06   3926           ; do assignment to set the watchdog timer
                                OF06   3927
                                09D5'CF    7F   OF06   3928      PUSHAQ   TIME_VALUE         ; location of the time
0FD B'CF    01   FB   OF0A   3929      CALLS    #1,TIM_ASG_RTN       ; do the assignment
                                OF0F   3930
                                000C   31   OF0F   3931      BRW      END_ASSIGN_SBR TN        ; and leave
                                OF12   3932
                                OF12   3933 PAR_ASG:
                                OF12   3934           ; do assignment to change the parallel count
                                OF12   3935
                                08B4'CF    7F   OF12   3936      PUSHAQ   ASSIGN_VALUE_DESC     ; the new count
101E'CF    01   FB   OF16   3937      CALLS    #1,PAR_ASG_RTN       ; do the assignment
                                OF1B   3938
                                0000   31   OF1B   3939      BRW      END_ASSIGN_SBR TN        ; and leave
                                OF1E   3940
                                OF1E   3941 END_ASSIGN_SBR TN:
                                OF1E   3942           ; all done with assignment, so leave
                                OF1E   3943
                                50    01   D0   OF1E   3944      MOVL     $$$ _NORMAL,R0        ; leave with success
                                OF21   3945
                                04    04   OF21   3946      RET              ; exit

```

```
00000004 0F22 3948 .SBTTL Assignment Routine for Process Name
0F22 3949
0F22 3950 ; define offsets for parameters
0F22 3951
0F22 3952 PNAME_ASG_DSC = 4
0F22 3953
0F22 3954 :++
0F22 3955 : FUNCTIONAL DESCRIPTION:
0F22 3956 :
0F22 3957 : This routine is called to change the current process name of the
0F22 3958 : TSTCNTRL.
0F22 3959 :
0F22 3960 : CALLING SEQUENCE:
0F22 3961 :
0F22 3962 : PUSHAQ ASSIGN_VALUE_DESC ; the new process name
0F22 3963 : CALLS #1,PNAME_ASG_RTN
0F22 3964 :
0F22 3965 : INPUT PARAMETERS:
0F22 3966 :
0F22 3967 : PNAME_ASG_DSC(AP) - address of a descriptor of the assignment value
0F22 3968 : (the process name)
0F22 3969 :
0F22 3970 : IMPLICIT INPUTS:
0F22 3971 :
0F22 3972 : None.
0F22 3973 :
0F22 3974 : OUTPUT PARAMETERS:
0F22 3975 :
0F22 3976 : None.
0F22 3977 :
0F22 3978 : IMPLICIT OUTPUTS:
0F22 3979 :
0F22 3980 : None.
0F22 3981 :
0F22 3982 : COMPLETION CODES:
0F22 3983 :
0F22 3984 : SS$_NORMAL
0F22 3985 :
0F22 3986 : SIDE EFFECTS:
0F22 3987 :
0F22 3988 : The process name will be changed.
0F22 3989 :--
0F22 3990
007C 0F22 3991 PNAME_ASG_RTN:
0F22 3992 .WORD ^M<R2,R3,R4,R5,R6>
0F24 3993
0F24 3994 ; put the new process name in the descriptor
0F24 3995
52 04 AC D0 0F24 3996 MOVL PNAME_ASG_DSC(AP),R2 ; get the address of new name
0F28 3997
0516'CF 62 B0 0F28 3998 MOVW DSC$W_LENGTH(R2),PRC_NAM_DESC ; get the new size
0F2D 3999
0F2D 4000 MOVCL DSC$W_LENGTH(R2),- ; and copy the new name in
0F2F 4001 @DSC$A_POINTER(R2),PRC_NAM_STR ; the descriptor
0F34 4002
0F34 4003 ; call the system service to set the process name
0F34 4004
```

TSTCNTRL
V04-000

TEST PACKAGE CONTROL PROGRAM
Assignment Routine for Process Name

D 8

16-SEP-1984 01:30:05
5-SEP-1984 04:26:03

VAX/VMS Macro V04-00
[UETP.SRC]UETPHAS00.MAR;1

Page 92
(44)

TS
V0

			0F34	4005	\$SETPRN_S	PRCNAM = PRC_NAM_DESC	; set the process name
			0F3F	4006			
50	01	D0	0F3F	4007	MOVL	#SS\$_NORMAL,R0	; return success
			0F42	4008			
		04	0F42	4009	RET		; exit

```
0F43 4011 .SBTTL Assignment Routine for Starting Sentinel
0F43 4012
0F43 4013 ; define offsets for parameters
0F43 4014
U0000004 0F43 4015 STRT_ASG_DSC = 4
0F43 4016
0F43 4017 :++
0F43 4018 : FUNCTIONAL DESCRIPTION:
0F43 4019 :
0F43 4020 : This routine is called to change the starting sentinel keyword used
0F43 4021 : when looking through the temporary log file.
0F43 4022 :
0F43 4023 : CALLING SEQUENCE:
0F43 4024 :
0F43 4025 : PUSHAD ASSIGN_VALUE_DESC ; the new sentinel
0F43 4026 : CALLS #1,PNAME_ASG_RTN
0F43 4027 :
0F43 4028 : INPUT PARAMETERS:
0F43 4029 :
0F43 4030 : STRT_ASG_DSC(AP) - address of a descriptor of the assignment value
0F43 4031 : (the process name)
0F43 4032 :
0F43 4033 : IMPLICIT INPUTS:
0F43 4034 :
0F43 4035 : None.
0F43 4036 :
0F43 4037 : OUTPUT PARAMETERS:
0F43 4038 :
0F43 4039 : None.
0F43 4040 :
0F43 4041 : IMPLICIT OUTPUTS:
0F43 4042 :
0F43 4043 : The value of START_DESC will be changed.
0F43 4044 :
0F43 4045 : COMPLETION CODES:
0F43 4046 :
0F43 4047 : SSS_NORMAL
0F43 4048 :
0F43 4049 : SIDE EFFECTS:
0F43 4050 :
0F43 4051 : None
0F43 4052 :--
0F43 4053 :
007C 0F43 4054 START_ASG_RTN:
0F43 4055 .WORD ^M<R2,R3,R4,R5,R6>
0F45 4056
0F45 4057 ; to make things simple, put the address of the descriptor in a regr.
0F45 4058
52 04 AC D0 0F45 4059 MOVL STRT_ASG_DSC(AP),R2 ; get the address of the desc.
0F49 4060
0F49 4061 ; now, move the assignment value into the actual sentinel descriptor
0F49 4062
028A'CF 62 B0 0F49 4063 MOVW DSC$W_LENGTH(R2),START_DESC ; put size in descriptor
62 28 0F4E 4064 MOVCL DSC$W_LENGTH(R2),- ; put in the new sentinel
0292'CF 04 B2 0F50 4065 @DSC$A_POINTER(R2),START_STR
0F55 4066
50 01 D0 0F55 4067 MOVL #SS$_NORMAL,R0 ; return success
```

TSTCNTRL
V04-000

F 8
TEST PACKAGE CONTROL PROGRAM 16-SEP-1984 01:30:05 VAX/VMS Macro V04-00
Assignment Routine for Starting Sentinel 5-SEP-1984 04:26:03 [UETP.SRC]UETPHAS00.MAR;1 Page 94
04 OF58 4068
04 OF58 4069 RET ; exit

TS
VC

```
00000004 0F59 4071 .SBTTL Assignment Routine for Ending Sentinel
0F59 4072
0F59 4073 ; define offsets for parameters
0F59 4074
0F59 4075 STP_ASG_DSC = 4
0F59 4076
0F59 4077 :++
0F59 4078 : FUNCTIONAL DESCRIPTION:
0F59 4079 :
0F59 4080 : This routine is called to change the stopping sentinel keyword used
0F59 4081 : when looking through the temporary log file.
0F59 4082 :
0F59 4083 : CALLING SEQUENCE:
0F59 4084 :
0F59 4085 : PUSHAQ ASSIGN VALUE_DESC ; the new sentinel
0F59 4086 : CALLS #1,PNAME_ASG_RTN
0F59 4087 :
0F59 4088 : INPUT PARAMETERS:
0F59 4089 :
0F59 4090 : STP_ASG_DSC(AP) - address of a descriptor of the assignment value
0F59 4091 :
0F59 4092 : IMPLICIT INPUTS:
0F59 4093 :
0F59 4094 : None.
0F59 4095 :
0F59 4096 : OUTPUT PARAMETERS:
0F59 4097 :
0F59 4098 : None.
0F59 4099 :
0F59 4100 : IMPLICIT OUTPUTS:
0F59 4101 :
0F59 4102 : The value of STOP_DESC will be changed.
0F59 4103 :
0F59 4104 : COMPLETION CODES:
0F59 4105 :
0F59 4106 : SSS_NORMAL
0F59 4107 :
0F59 4108 : SIDE EFFECTS:
0F59 4109 :
0F59 4110 : None
0F59 4111 :--
0F59 4112
007C 0F59 4113 STOP_ASG_RTN:
0F59 4114 .WORD ^M<R2,R3,R4,R5,R6>
0F5B 4115
0F5B 4116 ; to make things simple, put the address of the descriptor in a regr.
0F5B 4117
052 04 AC D0 0F5B 4118 MOVL STP_ASG_DSC(AP),R2 ; get the address of the desc.
0F5F 4119
0F5F 4120 ; now, move the assignment value into the actual sentinel descriptor
0F5F 4121
031B'CF 62 B0 0F5F 4122 MOVW DSC$W_LENGTH(R2),STOP_DESC ; put size in descriptor
0323'CF 04 B2 28 0F64 4123 MOVCL DSC$W_LENGTH(R2),- ; put in the new sentinel
0F66 4124 @DSC$A_POINTER(R2),STOP_STR
0F68 4125
050 01 D0 0F68 4126 MOVL #SS$_NORMAL,R0 ; return success
0F6E 4127
```

TSTCNTRL
V04-000

H 8
TEST PACKAGE CONTROL PROGRAM
Assignment Routine for Ending Sentinel
04 0F6E 4128 RET

16-SEP-1984 01:30:05 VAX/VMS Macro V04-00
5-SEP-1984 04:26:03 [UETP.SRC]UETPHAS00.MAR;1
; exit

Page 96
(46)

TS
VC

```
00000004 0F6F 4130 .SBTTL Assignment Routine Changing Comment Flag
0F6F 4131
0F6F 4132 ; define offsets for parameters
0F6F 4133
0F6F 4134 COM_ASG_DSC = 4
0F6F 4135
0F6F 4136 ;++
0F6F 4137 : FUNCTIONAL DESCRIPTION:
0F6F 4138 :
0F6F 4139 : This routine is called to change the current value of the flag which
0F6F 4140 : signals the TSTCNTRL to either print out comments or not.
0F6F 4141 :
0F6F 4142 : CALLING SEQUENCE:
0F6F 4143 :
0F6F 4144 : PUSHAQ ASSIGN_VALUE_DESC ; the comment string
0F6F 4145 : CALLS #1,COM_ASG_RTN
0F6F 4146 :
0F6F 4147 : INPUT PARAMETERS:
0F6F 4148 :
0F6F 4149 : COM_ASG_DSC(AP) - address of a descriptor of the the comment value
0F6F 4150 : (either YES or NO)
0F6F 4151 :
0F6F 4152 : IMPLICIT INPUTS:
0F6F 4153 :
0F6F 4154 : None.
0F6F 4155 :
0F6F 4156 : OUTPUT PARAMETERS:
0F6F 4157 :
0F6F 4158 : None.
0F6F 4159 :
0F6F 4160 : IMPLICIT OUTPUTS:
0F6F 4161 :
0F6F 4162 : May change the value of TCNTRL$V_PRNT_COMMENTS in FLAG
0F6F 4163 :
0F6F 4164 : COMPLETION CODES:
0F6F 4165 :
0F6F 4166 : SSS_NORMAL
0F6F 4167 :
0F6F 4168 : SIDE EFFECTS:
0F6F 4169 :
0F6F 4170 : Comments may or may not be printed out.
0F6F 4171 :--
0F6F 4172
000C 0F6F 4173 COM_ASG_RTN:
0F6F 4174 .WORD ^M<R2,R3>
0F71 4175
0F71 4176 ; put the address of the descriptor in a register to make things easy
0F71 4177
53 04 AC D0 0F71 4178 MOVL COM_ASG_DSC(AP),R3
0F75 4179 ; the keyword value is only allowed to be YES or NO
0F75 4180 ; move the address of the string into a register so that we may
0F75 4181 ; look at the first character easily
0F75 4182
52 04 B3 DE 0F75 4183 MOVAL @DSC$A_POINTER(R3),R2 ; get the address of the string
0F79 4184
0F79 4185 ; look at first character for a Y or N
0F79 4186
```

```
59 8F 62 91 0F79 4187      CMPB      (R2),#^A/Y/      ; is it yes?
                                0C 12 0F7D 4188      BNEQ      10$      ; no, try again
                                0F7F 4189
                                0F7F 4190      ; comments are to be printed, so turn flag on
                                0F7F 4191
04A3'CF 00000040 8F C8 0F7F 4192      BISL2      #TCNTRL$M_PRNT_COMMENTS,FLAGS ; yes, turn it on
                                0009 31 0F88 4193      BRW      END_COM_ASG_RTN ; all done
                                0F8B 4194
                                0F8B 4195 10$:
                                0F8B 4196      ; comments are not to be printed, so turn flag off
                                0F8B 4197
04A3'CF 00000040 8F CA 0F8B 4198      BICL2      #TCNTRL$M_PRNT_COMMENTS,FLAGS ; turn comments off
                                0F94 4199
                                0F94 4200 END_COM_ASG_RTN:
                                0F94 4201
                                50 01 D0 0F94 4202      MOVL      #$$$_NORMAL,R0      ; return success
                                0F97 4203
                                04 0F97 4204      RET
```

```
00000004 0F98 4206 .SBTTL Assignment Routine for Log File Name
0F98 4207
0F98 4208 ; define offsets for parameters
0F98 4209
0F98 4210 LGN_ASG_DSC = 4
0F98 4211
0F98 4212 :++
0F98 4213 : FUNCTIONAL DESCRIPTION:
0F98 4214 :
0F98 4215 : This routine is called to change the current name of the permanent
0F98 4216 : log file. The name can only be changed BEFORE the file gets created.
0F98 4217 : There is a flag (TCNTRL$V_SET_LOGNAM) which gets set when the file
0F98 4218 : is created.
0F98 4219
0F98 4220 : CALLING SEQUENCE:
0F98 4221 :
0F98 4222 : PUSHAQ FILE_SPEC_DESC ; the new logfile name
0F98 4223 : CALLS #1,LOG_ASG_RTN
0F98 4224 :
0F98 4225 : INPUT PARAMETERS:
0F98 4226 :
0F98 4227 : LOG_ASG_DSC(AP) - address of a descriptor of the assignment value
0F98 4228 : (the new log filename)
0F98 4229 :
0F98 4230 : IMPLICIT INPUTS:
0F98 4231 :
0F98 4232 : FLAGS - used to check if the name has already been set, and to
0F98 4233 : check to see if a error message is to be printed if it has.
0F98 4234 :
0F98 4235 : OUTPUT PARAMETERS:
0F98 4236 :
0F98 4237 : None.
0F98 4238 :
0F98 4239 : IMPLICIT OUTPUTS:
0F98 4240 :
0F98 4241 : None.
0F98 4242 :
0F98 4243 : COMPLETION CODES:
0F98 4244 :
0F98 4245 : $$$_NORMAL
0F98 4246 :
0F98 4247 : SIDE EFFECTS:
0F98 4248 :
0F98 4249 : The name of the log file may be changed.
0F98 4250 :--
0F98 4251
0004 0F98 4252 LOG_ASG_RTN:
0F98 4253 .WORD ^M<R2>
0F9A 4254
0F9A 4255 ; first check to see if the SET_LOGNAM flag has been set
0F9A 4256
1C 04A3'CF 07 E1 0F9A 4257 BBC #TCNTRL$V_SET_LOGNAM,FLAGS,20$ ; has not been set, so do assign
13 04A3'CF 0A E0 0FA0 4258 BBS #TCNTRL$V_REWOUND,FLAGS,10$ ; skip message if wrapping
0FA6 4259
0FA6 4260 ; it has been set, so print a warning saying that it can't be changed
0FA6 4261
03D9'CF DF 0FA6 4262 PUSHAL NO_SET_LOG ; say we can't change log name
```

```
00C31130 01 DD OFAA 4263      PUSHL #1
00000000'GF 8F DD OFAC 4264      PUSHL #TCNTRL$TEXT!STSSK_WARNING
03 FB OFB2 4265      CALLS #3,G^LIB$SIGNAL
      OFB9 4266
      OFB9 4267 10$:
      OFB9 4268      ; since it is too late to change the filename, just leave
      OFB9 4269
001B 31 OFB9 4270      BRW      END_LOG_ASG_RTN      ; if set, then skip it!
      OFBC 4271
      OFBC 4272 20$:
      OFBC 4273      ; we can change the filename, so put the new one in the filename buffer
      OFBC 4274      ; put the address of the filename descriptor in a register to make it easy
      OFBC 4275
      OFBC 4276
52 04 AC D0 OFBC 4277      MOVL      LGN_ASG_DSC(AP),R2      ; address of descriptor
      OFC0 4278
      OFC0 4279      ; now, update the old log filename descriptor
      OFC0 4280
      OFC0 4281
0173'CF 62 B0 OFC0 4281      MOVW      DSC$W_LENGTH(R2),LOG_FIL_DESC      ; put size in old descriptor
      OFC5 4282      MOVW      DSC$W_LENGTH(R2),-      ; put in new filename
017B'CF 04 B2 OFC7 4283      @DSC$A_POINTER(R2),LOG_FIL_STR
      OFCC 4284
      OFCC 4285      $FAB_STORE FAB = CNTRL_LOG_FAB,-      ; change the FAB
      OFCC 4286      FNS = LOG_FIL_DESC
      OFD7 4287
      OFD7 4288 END_LOG_ASG_RTN:
      OFD7 4289
50 01 D0 OFD7 4290      MOVL      #SS$_NORMAL,R0      ; return success
      OFDA 4291
      OFDA 4292      RET      ; and leave
```

```
00000004 0FDB 4294 .SBTTL Assignment Routine for Setting Watchdog Timer
0FDB 4295
0FDB 4296 ; define offsets for parameters
0FDB 4297
0FDB 4298 TIM_VAL = 4
0FDB 4299
0FDB 4300 :++
0FDB 4301 : FUNCTIONAL DESCRIPTION:
0FDB 4302 :
0FDB 4303 : This routine is called to set the watchdog timer which is used to
0FDB 4304 : prevent a process (or a group of processes) from hanging a number of
0FDB 4305 : TSTCNTRL tests to be run.
0FDB 4306 :
0FDB 4307 : CALLING SEQUENCE:
0FDB 4308 :
0FDB 4309 : PUSHAQ TIME_VALUE ; the complete system time
0FDB 4310 : CALLS #1,TIM_ASG_RTN
0FDB 4311 :
0FDB 4312 : INPUT PARAMETERS:
0FDB 4313 :
0FDB 4314 : TIM_VAL(AP) - address of the complete time in system format. The time
0FDB 4315 : has already been parsed out and converted from ascii
0FDB 4316 : to system.
0FDB 4317 :
0FDB 4318 : IMPLICIT INPUTS:
0FDB 4319 :
0FDB 4320 : FLAGS - FLAGS is checked to see if we should set the timer and to see
0FDB 4321 : see if would should an error message out.
0FDB 4322 :
0FDB 4323 : OUTPUT PARAMETERS:
0FDB 4324 :
0FDB 4325 : None.
0FDB 4326 :
0FDB 4327 : IMPLICIT OUTPUTS:
0FDB 4328 :
0FDB 4329 : FLAGS - If we set a timer the TCNTRL$V_TIMER_SET flag is set to
0FDB 4330 : indicated this.
0FDB 4331 :
0FDB 4332 : COMPLETION CODES:
0FDB 4333 :
0FDB 4334 : SS$_NORMAL
0FDB 4335 :
0FDB 4336 : SIDE EFFECTS:
0FDB 4337 :
0FDB 4338 : It will cause a timer to be set and active. This timer may cause
0FDB 4339 : the entire TSTCNTRL to abort if expired.
0FDB 4340 :--
0FDB 4341 :
000C 0FDB 4342 TIM_ASG_RTN:
0FDB 4343 .WORD ^M<R2,R3>
0FDD 4344
0FDD 4345 ; if timer is currently set, then print a warning, otherwise
0FDD 4346 ; just do assignment
0FDD 4347
1C 04A3'CF 08 E1 0FDD 4348 BBC #TCNTRL$V_TIMER_SET,FLAGS,20$ ; not set, do assignment
13 04A3'CF 0A E0 0FE3 4349 BBS #TCNTRL$V_REWOUND,FLAGS,10$ ; skip message if wrapping
0FE9 4350
```

```
02E0'CF 7F 0FE9 4351      PUSHQ  NO_SET_TIM      ; say that we can't set it
          01 DD 0FED 4352      PUSHL  #1
00C31130 8F DD 0FEF 4353      PUSHL  #TCNTRL$TEXT!ST$K_WARNING
00000000'GF 03 FB 0FF5 4354      CALLS  #3,G^LIB$SIGNAL
          03 0FFC 4355
          03 0FFC 4356 10$:
          03 0FFC 4357      ; we can't set the timer while one is active, so leave
          03 0FFC 4358
001B      31 0FFC 4359      BRW      END_TIM_ASG_RTN
          03 0FFF 4360
          03 0FFF 4361 20$:
          03 0FFF 4362      ; a timer is not currently running, so we are able to start one.
          03 0FFF 4363
04A3'CF 00000100 8F C8 0FFF 4364      BISL2  #TCNTRL$M_TIMER_SET,FLAGS      ; say that a timer is active
          08 1008 4365
          08 1008 4366      $SETIMR_S DAYTIM = @TIM_VAL(AP),-      ; start the timer
          08 1008 4367      ASTADR = TIME_EXP_AST,-      ; if expire, go here
          08 1008 4368      REQIDT = #MAXTIME_ID      ; timer id is 1
          08 101A 4369
          08 101A 4370
          08 101A 4371 END_TIM_ASG_RTN:
          08 101A 4372
          50 01 D0 101A 4373      MOVL  #SS$_NORMAL,R0      ; return success
          04 101D 4374
          04 101D 4375      RET
```

```
00000004 101E 4377 .SBTTL Assignment Routine for Changing the Parallel Count
101E 4378
101E 4379 ; define offsets for parameters
101E 4380
101E 4381 PAR_ASG_DSC = 4
101E 4382
101E 4383 :++
101E 4384 : FUNCTIONAL DESCRIPTION:
101E 4385 :
101E 4386 : This routine is called to change the value of the parallel count
101E 4387 : (that is, the number of processes allowed to execute in parallel).
101E 4388 : It will make sure that no processes are currently executing, and,
101E 4389 : if this is true, will change the count for the next set.
101E 4390 :
101E 4391 : CALLING SEQUENCE:
101E 4392 :
101E 4393 : PUSHAQ ASSIGN_VALUE_DESC ; the new count
101E 4394 : CALLS #1,PAR_ASG_RTN
101E 4395 :
101E 4396 : INPUT PARAMETERS:
101E 4397 :
101E 4398 : PAR_ASG_DSC(AP) - address of a descriptor of the assignment value
101E 4399 : (the new count in ASCII)
101E 4400 :
101E 4401 : IMPLICIT INPUTS:
101E 4402 :
101E 4403 : None.
101E 4404 :
101E 4405 : OUTPUT PARAMETERS:
101E 4406 :
101E 4407 : None.
101E 4408 :
101E 4409 : IMPLICIT OUTPUTS:
101E 4410 :
101E 4411 : The value of TOTAL_PROC_LIMIT and ACTIVE_PROC_LIMIT will be changed.
101E 4412 :
101E 4413 : COMPLETION CODES:
101E 4414 :
101E 4415 : SSS_NORMAL
101E 4416 :
101E 4417 : SIDE EFFECTS:
101E 4418 :
101E 4419 : None
101E 4420 :--
101E 4421 :
0000 101E 4422 PAR_ASG_RTN:
101E 4423 .WORD ^M<>
1020 4424
1020 4425 :
1020 4426 : First check to see if any processes are currently running. If some are,
1020 4427 : then print out a warning and ignore the assignment.
1020 4428 :
1020 4429 :
0000'CF D5 1020 4430 TSTL PROCESS_RUNNING ; are any processes running
1F 13 1024 4431 BEQL 10$ ; no, then OK to do assignment
1026 4432
1026 4433 ;
```

```
1026 4434 ; Processes are still running, so unable to perform assignment,
1026 4435 ; print out a warning.
1026 4436 ;
1026 4437 ;
03 04A3'CF 0A E1 1026 4438 BBC #TCNTRL$V_REWOUND,FLAGS,5$ ; if not wrapping, then write msg
004A 31 102C 4439 BRW 30$ ; if wrapping, just ignore asg
102F 4440
102F 4441 5$:
0394'CF 7F 102F 4442 PUSHAQ PROC_ACT ; yes, print warning . . .
01 DD 1033 4443 PUSHL #1
00C31130 8F DD 1035 4444 PUSHL #TCNTRL$ TEXT!STSSK_WARNING
00000000'GF 03 FB 103B 4445 CALLS #3,G^LIB$SIGNAL
1042 4446
0034 31 1042 4447 BRW 30$
1045 4448
1045 4449 10$:
1045 4450
1045 4451 ;
1045 4452 ; Everything is OK, so perform the assignment.
1045 4453 ;
1045 4454 ;
1045 4455 ;
1045 4456 ; Convert assignment value into a number
1045 4457 ;
1045 4458 ;
000C'CF DF 1045 4459 PUSHAL TOTAL_PROC_LIMIT ; place for converted number
04 BC 7F 1049 4460 PUSHAQ @PAR_ASG_DSC(AP) ; string to be converted
00000000'GF 02 FB 104C 4461 CALLS #2,G^OTSS$CVT_TI_L ; convert text to longword
1053 4462
1053 4463 ;
1053 4464 ; See if valid PARCNT
1053 4465 ;
1053 4466 ;
00 000C'CF D1 1053 4467 CMPL TOTAL_PROC_LIMIT,#0 ; is PARCNT <= 0?
18 14 1058 4468 BGTR 20$ ; no, continue
0355'CF DF 105A 4469 PUSHAL ZERO_PARCNT ; yes, print warning . . .
01 DD 105E 4470 PUSHL #1
00C31130 8F DD 1060 4471 PUSHL #TCNTRL$ TEXT!STSSK_WARNING
00000000'GF 03 FB 1066 4472 CALLS #3,G^LIB$SIGNAL
106D 4473
000C'CF 01 D0 106D 4474 MOVL #1,TOTAL_PROC_LIMIT ; . . . and use PARCNT = 1
1072 4475
1072 4476 20$:
000C'CF D0 1072 4477 MOVL TOTAL_PROC_LIMIT -
0008'CF 1076 4478 ACTIVE_PROC_LIMIT ; set maximum number of
1079 4479 ; active processes
1079 4480 30$:
50 01 D0 1079 4481 MOVL #SS$_NORMAL,R0 ; return success
04 107C 4482 RET
107D 4483
```

```
107D 4485 .SBTTL Get and Parse Record
107D 4486 :++
107D 4487 : FUNCTIONAL DESCRIPTION:
107D 4488 :
107D 4489 : This routine is called whenever a new input record is required. It
107D 4490 : will perform a GET to fetch the record and will then parse the record
107D 4491 : out into its various components setting parse flags to signal the type
107D 4492 : of record which was just fetched.
107D 4493 :
107D 4494 : CALLING SEQUENCE:
107D 4495 :
107D 4496 : CALLS #0,GET_PARSE_SBRTN
107D 4497 :
107D 4498 : INPUT PARAMETERS:
107D 4499 :
107D 4500 : None.
107D 4501 :
107D 4502 : IMPLICIT INPUTS:
107D 4503 :
107D 4504 : FLAGS -- the TSTCNTRL flags
107D 4505 : CONTROL_CASE
107D 4506 : PARSE_FLAGS
107D 4507 : The data file RMS structures are used to fetch the next data record.
107D 4508 :
107D 4509 : OUTPUT PARAMETERS:
107D 4510 :
107D 4511 : None.
107D 4512 :
107D 4513 : IMPLICIT OUTPUTS:
107D 4514 :
107D 4515 : All of the parse fields may be changed. (See the parse read/write area
107D 4516 : which describes all these fields.)
107D 4517 :
107D 4518 : COMPLETION CODES:
107D 4519 :
107D 4520 : $$$_NORMAL
107D 4521 :
107D 4522 : SIDE EFFECTS:
107D 4523 :
107D 4524 : May change the values of some the flags.
107D 4525 :--
107D 4526 GET_PARSE_SBRTN:
OFFC 107D 4527 .WORD ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
107D 4528
107F 4529 ; initialized the stack area for local storage
107F 4530
107F 4531 SUBL2 #GET_PARSE_LENGTH,SP ; allocate space for local storage
1086 4532 MOVCS #0,#0,#0,#GET_PARSE_LENGTH,(SP) ; clear out local storage
108F 4533 MOVL SP,R11 ; keep pointer to local area
1092 4534
1092 4535 GET_DAT_RCRD:
1092 4536 ; ready to get a record and set it up to be parsed
1092 4537
1092 4538 ; clear out the parse flags since we are working on a new record
1092 4539
1092 4540 CLRL PARSE_FLAGS ; reset flags from previous parse
0659'CF D4 1092 4541 CLRL ASSIGN_CASE ; reset assignment case flags
04A7'CF D4 1096
```

```
6E 0254 8F 5E 00000254 8F 02
00 00 8F 00 2C
5B 00 D0
```

```
049F'CF D4 109A 4542 CLRL CREATE_PROC_CASE ; reset case for creating proc
109E 4543
109E 4544 ; get the record from the input data file and put it in a buffer from
109E 4545 ; which we can work
109E 4546
109E 4547 $GET RAB = DATA_FILE_RAB,- ; use the input file RAB
109E 4548 ERR = RMSEERR
10AD 4549
10AD 4550 ; see if we got a record or if we are at E-O-F
10AD 4551
50 00000000'8F D1 10AD 4552 CMPL #RMSE_EOF,R0 ; is status end-of-file?
08 12 10B4 4553 BNEQ 10$ ; nc, we have a record, continue
10B6 4554
10B6 4555 ; we are at end-of-file, so set a flag and skip the parse
10B6 4556
049B'CF 08 C8 10B6 4557 BISL2 #CASE$M_EOF,CONTROL_CASE ; at end-of-file
0040 31 10BB 4558 BRW CHK_END_SEG ; check for end-of-segment
10BE 4559
10BE 4560 10$:
0B02'CF B0 10BE 4561 MOVW <DATA_FILE_RAB+RAB$W_RSZ>,- ; put the size of the input string
065D'CF 10C2 4562 <DATA_RCRD_DESC+DSC$W_LENGTH> ; in the record descriptor
10C5 4563
10C5 4564 ; now, upcase the entire string so we don't have to worry about matches
10C5 4565
065D'CF 7F 10C5 4566 PUSHAQ DATA_RCRD_DESC ; change the input string
065D'CF 7F 10C9 4567 PUSHAQ DATA_RCRD_DESC ; to become
00000000'GF 02 FB 10CD 4568 CALLS #2,G*STR$OPCASE ; all caps
10D4 4569
10D4 4570 ; Ok, we now have a record in the proper format, so call TPARSE to
10D4 4571 ; parse it out into its components
10D4 4572
10D4 4573 ; initialize the TPARSE parameter block
10D4 4574
08 D0 10D4 4575 MOVL #TPASK_COUNT0,- ; put count in the
68 10D6 4576 TPASL_COUNT+DAT_PARSE_BLK(R11) ; TPARSE block
10D7 4577
04 AB D4 10D7 4578 CLRL TPASL_OPTIONS+DAT_PARSE_BLK(R11) ; clear the options field
02 C8 10DA 4579 BISL2 #TPAS$M_ABBREV,- ; allow abbreviations
04 AB 10DC 4580 TPASL_OPTIONS+DAT_PARSE_BLK(R11)
10DE 4581
10DE 4582 PARSE_DAT_RCRD:
10DE 4583
10DE 4584 ; now copy the descriptor of the data record into the TPARSE block so
10DE 4585 ; that it knows what it is looking at.
10DE 4586
065D'CF D0 10DE 4587 MOVL <DATA_RCRD_DESC+DSC$W_LENGTH>,- ; put the length in the TPARSE
08 AB 10E2 4588 TPASL_STRINGCNT+DAT_PARSE_BLK(R11) ; count field
10C4 4589
0661'CF D0 10E4 4590 MOVL <DATA_RCRD_DESC+DSC$A_POINTER>,- ; put the address in the TPARSE
0C AB 10E8 4591 TPASL_STRINGPTR+DAT_PARSE_BLK(R11) ; address field
10EA 4592
10EA 4593 ; finally call TPARSE
10EA 4594
10EA 4595
0000'CF DF 10EA 4596 PUSHAL DAT_FILE_KEY ; use the data file keys
0000'CF DF 10EE 4597 PUSHAL DAT_FILE_STATE ; and the data file state table
68 DF 10F2 4598 PUSHAL DAT_PARSE_BLK(R11) ; and the data file param. block
```

```
00000000'GF 03 FB 10F4 4599 CALLS #3,G^LIB$TPARSE ; perform the parse
                  10FB 4600
                  10FB 4601 ; parse is done, did it succeed without syntax errors?
                  10FB 4602
47 50 E9 10FB 4603 BLBC R0,SYNTAX_ERROR ; there was an error, so report it
                  10FE 4604
                  10FE 4605 CHK_END_SEG:
                  10FE 4606
                  10FE 4607 ; now, see if the watchdog timer expired. If it did, then continue
                  10FE 4608 ; to get and parse records until the end of a segment is reached. The
                  10FE 4609 ; end of a segment is either an end-of-file, or a process to be run
                  10FE 4610 ; sequentially.
                  10FE 4611
                  10FE 4612 ; first see if the timer expired
                  10FE 4613
35 04A3'CF 0E E1 10FE 4614 BBC #TCNTRL$V_TIME_EXP,FLAGS,GET_AGAIN ; did the timer expire?
                  1104 4615
                  1104 4616 ; maxtime is set, so get ready to get next record and see if at EOF
                  1104 4617
0659'CF 02 C8 1104 4618 BISL2 #PARSES$M_GET_NXT_RCRD,PARSE_FLAGS ; prepare to get another record
10 049B'CF 03 E1 1109 4619 BBC #CASE$V_EOF,CONTROL_CASE,10$ ; not EOF, keep checking
                  110F 4620
                  110F 4621 ; EOF is set, so end-of-segment, stop gets
                  110F 4622
0659'CF 02 CA 110F 4623 BICL2 #PARSES$M_GET_NXT_RCRD,PARSE_FLAGS ; stop gets
04A3'CF 00004000 8F CA 1114 4624 BICL2 #TCNTRL$M_TIME_EXP,FLAGS ; reset max time flag
                  111D 4625
                  111D 4626 BRB GET_AGAIN ; continue
1A 11 111D 4627
                  111F 4628
                  111F 4629 10$:
                  111F 4630 ; are we at a filespec?
                  111F 4631
14 049B'CF 04 E1 111F 4632 BBC #CASE$V_FILE,CONTROL_CASE,GET_AGAIN ; no filespec, keep trying
                  1125 4633
                  1125 4634 ; we have a filespec, is it sequential?
                  1125 4635
00 E0 1125 4636 BBS #PARSES$V_PROC_RUNS_OTHERS,- ; not sequential, keep trying
OE 0659'CF 1127 4637 PARSE_FLAGS,GET_AGAIN
                  112B 4638
                  112B 4639 ; it is sequential, end-of-segment
                  112B 4640
0659'CF 02 CA 112B 4641 BICL2 #PARSES$M_GET_NXT_RCRD,PARSE_FLAGS ; stop gets
04A3'CF 00004000 8F CA 1130 4642 BICL2 #TCNTRL$M_TIME_EXP,FLAGS ; reset max time flag
                  1139 4643
                  1139 4644 GET_AGAIN:
                  1139 4645
                  1139 4646 ; the parse succeeded, but we may want to get another record
                  1139 4647
03 0659'CF 01 E0 1139 4648 BBS #PARSES$V_GET_NXT_RCRD,PARSE_FLAGS,10$ ; see if we need a record
00CA 31 113F 4649 BRW END_GET_PARSE ; we don't so exit
                  1142 4650
                  1142 4651 10$:
                  1142 4652 ; the parse determined that we are to get another record, so start over
                  1142 4653
FF4D 31 1142 4654 BRW GET_DAT_RCRD ; get another record
                  1145 4655
```

```
1145 4656 SYNTAX_ERROR:
1145 4657
1145 4658 ; there was a syntax error in the record, write out a message saying
1145 4659 ; what record failed and what caused the error
1145 4660
55 50 D0 1145 4661 MOVL R0,R5 ; save the status
1148 4662
1148 4663 ; make the descriptor of the record in error
1148 4664
1148 4665 ; first initialize the descriptors
1148 4666
34 AB DE 1148 4667 MOVAL RCRD_ERR_STR(R11),- ; put the address in the
28 AB 1148 4668 DSC$A_POINTER+RCRD_ERR_DESC(R11) ; descriptor
114D 4669
34 AB DE 114D 4670 MOVAL RCRD_ERR_STR(R11),- ; put the address in the
30 AB 1150 4671 DSC$A_POINTER+RCRD_ERR_FAO(R11) ; descriptor
0108 8F 3C 1152 4672 MOVZWL #<RCRD_SIZE*2>,- ; put the size in the
2C AB 1156 4673 DSC$W_LENGTH+RCRD_ERR_FAO(R11) ; descriptor
1158 4674
1158 4675 ; then make the error message
1158 4676
52 065D'CF D0 1158 4677 MOVL DSC$W_LENGTH+DATA_RCRD_DESC,R2 ; get size for the FAO
53 0661'CF D0 115D 4678 MOVL DSC$A_POINTER+DATA_RCRD_DESC,R3 ; get address for the FAO
1162 4679
1162 4680 $FAO_S CTRSTR = RCRD_ERR_MSG,- ; the error message
1162 4681 OUTLEN = DSC$W_LENGTH+RCRD_ERR_DESC(R11),- ; the output length
1162 4682 OUTBUF = RCRD_ERR_FAO(R11),- ; the output string
1162 4683 P1 = R2,- ; length of invalid record
1162 4684 P2 = R3 ; address of invalid record
1162 4685
1177 4686
1177 4687 ; now see if we have a TPARSE error (which implies we are pointing
1177 4688 ; to the invalid token), or some other type of error (we just print
1177 4689 ; out the error message).
1177 4690
00158284 8F 55 D1 1177 4691 CMPL R5,#LIB$_SYNTAXERR ; is it a TPARSE syntax error?
03 12 117E 4692 BNEQ 10$ ; no, it is another type
1180 4693
0021 31 1180 4694 BRW TPARSE_ERR ; it's a TPARSE error, write
1183 4695 ; the token
1183 4696
1183 4697 10$:
1183 4698 ; the syntax error was not found by TPARSE, so we don't have a token.
1183 4699 ; instead, we will print out the invalid line, and the error message.
1183 4700
55 DD 1183 4701 PUSHL R5 ; write the error message
24 AB 7F 1185 4702 PUSHAQ RCRD_ERR_DESC(R11) ; and the invalid record
01 DD 1188 4703 PUSHL #1
00C31130 8F DD 118A 4704 PUSHL #TCNTRL$TEXT!ST$K_WARNING
00000000'GF 04 FB 1190 4705 CALLS #4,G^LIB$SIGNAL ; output the message
1197 4706
1197 4707 ; since parsed failed, get another record and try again
1197 4708
1197 4709 ; we may have set control bits before the parse failed. clear out
1197 4710 ; those bits and start again
1197 4711
049B'CF 10 CA 1197 4712 BICL2 #CASE$M_FILE.CONTROL_CASE ; clear file bit (may be set)
```

```
049B'CF 20 CA 119C 4713 BICL2 #CASE$M_ASSIGN,CONTROL_CASE ; clear assign bit (may be set)
      FEE 31 11A1 4714
      11A1 4715 BRW GET_DAT_RCRD ; get a new record
      11A4 4716
      11A4 4717
      11A4 4718 TPARSE_ERR:
      11A4 4719 ; the error was found by TPARSE, therefore we have an invalid token.
      11A4 4720 ; we will write out the invalid string along with the invalid token.
      11A4 4721
      11A4 4722
      11A4 4723 ; initialize the descriptors to do the FAO
      11A4 4724
      014C CB DE 11A4 4725 MOVAL TOKEN_ERR_STR(R11),- ; put the address in the
      0140 CB 11A8 4726 DSC$A_POINTER+TOKEN_ERR_DESC(R11) ; descriptor
      11AB 4727
      11AB 4728
      014C CB DE 11AB 4729 MOVAL TOKEN_ERR_STR(R11),- ; put the address in the
      0148 CB 11AF 4730 DSC$A_POINTER+TOKEN_ERR_FAO(R11) ; descriptor
      0108 8F 3C 11B2 4731 MOVZWL #<RCRD_SIZE*2>,- ; put the size in the
      0144 CB 11B6 4732 DSC$W_LENGTH+TOKEN_ERR_FAO(R11) ; descriptor
      11B9 4733
      11B9 4734 ; make error message of the invalid token
      11B9 4735
      52 10 AB DO 11B9 4736 MOVL TPASL_TOKENCNT+DAT_PARSE_BLK(R11),R2 ; get size for the FAO
      53 14 AB DO 11BD 4737 MOVL TPASL_TOKENPTR+DAT_PARSE_BLK(R11),R3 ; get address for the FAO
      11C1 4738
      11C1 4739 $FAO_S CTRSTR = TOKEN_ERR_MSG,- ; the error message
      11C1 4740 OUTLEN = DSC$W_LENGTH+TOKEN_ERR_DESC(R11),- ; the output length
      11C1 4741 OUTBUF = TOKEN_ERR_FAO(R11),- ; the output string
      11C1 4742 P1 = R2,- ; length of invalid token
      11C1 4743 P2 = R3 ; address of invalid token
      11D8 4744
      11D8 4745 ; now write out the error
      11D8 4746
      013C 55 DD 11D8 4747 PUSHL R5 ; TPARSE's complaint
      01 7F 11DA 4748 PUSHAQ TOKEN_ERR_DESC(R11) ; the invalid token
      00C31133 01 DD 11DE 4749 PUSHL #1
      8F DD 11E0 4750 PUSHL #TCNTRL$TEXT!STSSK_INFO
      11E6 4751
      10 009E'CF FO 11E6 4752 INSV FAC_CODE,#STSSV_FAC_NO,- ; set the facility code of the
      0C 11EB 4753 #STSSS_FAC_NO,- ; message to be the defined
      6E 11EC 4754 (SP) ; facility code
      11ED 4755
      24 AB 7F 11ED 4756 PUSHAQ RCRD_ERR_DESC(R11) ; the invalid record
      01 DD 11F0 4757 PUSHL #1
      00C31130 8F DD 11F2 4758 PUSHL #TCNTRL$TEXT!STSSK_WARNING
      00000000'GF 07 FB 11F8 4759 CALLS #7,G^LIB$SIGNAL ; write the message
      11FF 4760
      11FF 4761 ; now, since the parse failed, get another record and try again
      11FF 4762
      11FF 4763 ; we may have set control bits before the parse failed. clear out
      11FF 4764 ; those bits and start again
      11FF 4765
      049B'CF 10 CA 11FF 4766 BICL2 #CASE$M_FILE,CONTROL_CASE ; clear file bit (may be set)
      049B'CF 20 CA 1204 4767 BICL2 #CASE$M_ASSIGN,CONTROL_CASE ; clear assign bit (may be set)
      1209 4768
      FE86 31 1209 4769 BRW GET_DAT_RCRD ; get a new record
```

TSTCNTRL
V04-000

TEST PACKAGE CONTROL PROGRAM
Get and Parse Record

I 9

16-SEP-1984 01:30:05 VAX/VMS Macro V04-00
5-SEP-1984 04:26:03 [UETP.SRC]UETPHAS00.MAR;1

Page 110
(51)

TS
VC

```
120C 4770  
120C 4771 END_GET_PARSE:  
120C 4772  
120C 4773 ; all done parsing, exit with success  
120C 4774  
50 01 D0 120C 4775 MOVL #SS$ _NORMAL, R0 ; successful exit  
120F 4776  
04 120F 4777 RET
```

```
1210 4779 .SBTTL State Table for Data File
1210 4780
1210 4781 $INIT_STATE DAT_FIL_STATE,DAT_FIL_KEY
1210 4782
1210 4783 ; start state. look for a keyword, a comment, or a blank line,
1210 4784 ; anything else is an error, so signal it and get the next line.
1210 4785
1210 4786 $STATE START
1210 4787 $STRAN 'Y*',RUN PROC,,CASE$M_FILE,CONTROL CASE
1210 4788 $STRAN 'N*',TPAS_EXIT,,PARSE$M_GET_NXT_RCRD,PARSE_FLAGS
1210 4789 $STRAN 'LOG',LOGFIL ASG,,ASSIGN$M_LOG,ASSIGN CASE
1210 4790 $STRAN 'NAME',GENERIC ASG,,ASSIGN$M_NAME,ASSIGN CASE
1210 4791 $STRAN 'START',GENERIC ASG,,ASSIGN$M_START,ASSIGN CASE
1210 4792 $STRAN 'STOP',GENERIC ASG,,ASSIGN$M_STOP,ASSIGN CASE
1210 4793 $STRAN 'COMMENTS',GENERIC ASG,,ASSIGN$M_COM,ASSIGN CASE
1210 4794 $STRAN 'MAXTIME',TIME ASG,,ASSIGN$M_TIME,ASSIGN CASE
1210 4795 $STRAN 'PARALLEL',GENERIC ASG,,ASSIGN$M_PAR,ASSIGN CASE
1210 4796 $STRAN '!',TPAS_EXIT,,PARSE$M_GET_NXT_RCRD,PARSE_FLAGS
1210 4797 $STRAN 'EXIT',TPAS_EXIT,,CASE$M_EOF,CONTROL CASE
1210 4798 $STRAN TPAS_EOS,TPAS_EXIT,,PARSE$M_GET_NXT_RCRD,PARSE_FLAGS
1210 4799
1210 4800
1210 4801 ; get ready to parse the filespec, but first see if the process can
1210 4802 ; run with others, or if it must run alone. PARSE$V_PROC_RUNS_OTHERS
1210 4803 ; is set to be zero (meaning that it doesn't run with others) unless
1210 4804 ; it is changed here.
1210 4805
1210 4806 $STATE RUN PROC
1210 4807 $STRAN 'Y*',CHK_FILESPC,,PARSE$M_PROC_RUNS_OTHERS,PARSE_FLAGS
1210 4808 $STRAN 'N*',CHK_FILESPC
1210 4809
1210 4810 ; see if there is a filespec there before we try and parse one. Null
1210 4811 ; filespecs are legal syntax, but we flag that it is null.
1210 4812
1210 4813 $STATE CHK_FILESPC
1210 4814 $STRAN '!',TPAS_EXIT,,CRTPRC$M_NULL,CREATE_PROC_CASE
1210 4815 $STRAN TPAS_EOS,TPAS_EXIT,,CRTPRC$M_NULL,CREATE_PROC_CASE
1210 4816 $STRAN TPAS_LAMBDA,,EXPLICIT_BLANKS
1210 4817
1210 4818 ; with blanks on, check for the blank before the filespec
1210 4819
1210 4820 $STATE
1210 4821 $STRAN TPAS_BLANK,FIND_FILESPEC
1210 4822
1210 4823 ; we are to the point where a filespec better exists or the syntax is
1210 4824 ; wrong. we have set TPARSE so that it looks at ever blank (which we
1210 4825 ; will use as a delimiter unless it is in a quoted string). we will now
1210 4826 ; take apart what we think is a filespec and send that string to RMS
1210 4827 ; $PARSE to confirm whether or not it is a filespec.
1210 4828
1210 4829 $STATE FIND_FILESPEC
1210 4830 $STRAN !FILE_SPEC,,IMPLICIT_BLANKS,,SPEC_DESC
1210 4831
1210 4832 ; we have a string which we think is a filespec, call action routine
1210 4833 ; which uses $PARSE to confirm it.
1210 4834
1210 4835 $STATE
```

1210	4836	\$STRAN	TPAS_LAMBDA,FINISH_LINE,PRS_FILE_SPEC	
1210	4837			
1210	4838			
1210	4839			
1210	4840	\$STATE	FINISH_LINE	
1210	4841	\$STRAN	'!';TPAS_EXIT,GET_COMMENT	; get associated comment
1210	4842	\$STRAN	TPAS_EOS,TPAS_EXIT	
1210	4843	\$STRAN	TPAS_ANY,GET_PARAM,,,DELIMIT	
1210	4844			

```
1210 4846 ; states to parse out the parameter string. the delimiters may be any
1210 4847 ; character except the exclamation point '!'. when we get here the
1210 4848 ; first delimiter is in DELIMIT
1210 4849
1210 4850 $STATE GET_PARAM
1210 4851 $STRAN TPAS_LAMBDA,,EXPLICIT_BLANKS
1210 4852
1210 4853 ; while accepting blanks, take every character as being valid. the
1210 4854 ; action routine copies the parameter list into a buffer
1210 4855
1210 4856 $STATE
1210 4857 $STRAN !PARAM_STRING,,COPY_PARAM
1210 4858
1210 4859 ; accept the last character (it is the ending delimiter) and ignore
1210 4860 ; blanks again
1210 4861
1210 4862 $STATE
1210 4863 $STRAN TPAS_ANY,,IMPLICIT_BLANKS
1210 4864
1210 4865 ; see if we have a comment, else leave
1210 4866
1210 4867 $STATE
1210 4868 $STRAN '!',TPAS_EXIT,GET_COMMENT
1210 4869 $STRAN TPAS_EOS,TPAS_EXIT
1210 4870 $STRAN TPAS_LAMBDA,,TPAS_FAIL
1210 4871
1210 4872 ; this is the subexpression to accept every character which may be in
1210 4873 ; the parameter string. it calls an action routine which checks to
1210 4874 ; see if the character is a delimiter or not.
1210 4875
1210 4876 $STATE PARAM_STRING
1210 4877 $STRAN TPAS_EOS,TPAS_FAIL
1210 4878 $STRAN TPAS_ANY,PARAM_STRING,TST_DELIM
1210 4879 $STRAN TPAS_LAMBDA,TPAS_EXIT
```

```
1210 4881 ; subexpression to get the file spec string. successful completion
1210 4882 ; of this subexpression does not necessarily guarantee that the string
1210 4883 ; is indeed a filespec. this routine pulls out characters that may be
1210 4884 ; a filespec and then sends them to $PARSE to make sure.
1210 4885
1210 4886 $STATE FILE_SPEC
1210 4887 $STRAN '""',Q_STRING
1210 4888 $STRAN !NON_DELIMITER
1210 4889 $STRAN TPAS_LAMBDA,TPAS_EXIT
1210 4890
1210 4891 $STATE
1210 4892 $STRAN TPAS_ANY,FILE_SPEC ; collect all non-delimiters
1210 4893
1210 4894 ; see if we have a valid quoted string
1210 4895
1210 4896 $STATE Q_STRING
1210 4897 $STRAN '""',GET_REST
1210 4898 $STRAN TPAS_ANY,Q_STRING
1210 4899
1210 4900 ; now, collect the rest of the supposed filespec
1210 4901
1210 4902 $STATE GET_REST
1210 4903 $STRAN '""',TPAS_FAIL ; only 1 pair of quotes allowed
1210 4904 $STRAN !NON_DELIMITER
1210 4905 $STRAN TPAS_LAMBDA,TPAS_EXIT
1210 4906
1210 4907 $STATE
1210 4908 $STRAN TPAS_ANY,GET_REST
1210 4909
1210 4910 ; subexpression to determine whether the token is a valid delimiter
1210 4911 ; or not
1210 4912
1210 4913 $STATE NON_DELIMITER
1210 4914 $STRAN TPAS_EOS,TPAS_FAIL
1210 4915 $STRAN '!',TPAS_FAIL
1210 4916 $STRAN TPAS_BLANK,TPAS_FAIL
1210 4917 $STRAN TPAS_LAMBDA,TPAS_EXIT ; anything else is a non-delimiter
```

```
1210 4919      ; state tables for the various assignment statements
1210 4920
1210 4921      ; we have an assignment to change the name of the logfile.  parse
1210 4922      ; out the filename
1210 4923
1210 4924      $STATE LOGFIL_ASG
1210 4925      $STRAN '=',,EXPLICIT_BLANKS,CASE$M_ASSIGN,CONTROL_CASE
1210 4926
1210 4927      $STATE
1210 4928      $STRAN TPAS_BLANK
1210 4929      $STRAN TPAS_EOS,TPAS_FAIL
1210 4930      $STRAN TPAS_LAMBDA
1210 4931
1210 4932      $STATE
1210 4933      $STRAN !FILE_SPEC,,IMPLICIT_BLANKS,,SPEC_DESC
1210 4934
1210 4935      ; got the filespec, now $PARSE it and put it in the right place and
1210 4936      ; ignore the rest of the line
1210 4937
1210 4938      $STATE
1210 4939      $STRAN TPAS_LAMBDA,TPAS_EXIT,PRS_FILE_SPEC
1210 4940
1210 4941
1210 4942
1210 4943      ; we have a generic assignment, so just parse out the string and
1210 4944      ; put it in the proper buffer
1210 4945
1210 4946      $STATE GENERIC_ASG
1210 4947      $STRAN '=',,CASE$M_ASSIGN,CONTROL_CASE
1210 4948
1210 4949      $STATE
1210 4950      $STRAN TPAS_SYMBOL,TPAS_EXIT,GET_ASG_VAL
1210 4951
1210 4952      ; we have an assignment for the time, so parse out the time and
1210 4953      ; convert it to system time
1210 4954
1210 4955      $STATE TIME_ASG
1210 4956      $STRAN '=',,EXPLICIT_BLANKS,CASE$M_ASSIGN,CONTROL_CASE
1210 4957
1210 4958      $STATE
1210 4959      $STRAN TPAS_BLANK
1210 4960      $STRAN TPAS_EOS,TPAS_FAIL
1210 4961      $STRAN TPAS_LAMBDA
1210 4962
1210 4963
1210 4964      $STATE
1210 4965      $STRAN !GET_TIME_STRING,,IMPLICIT_BLANKS,,TIME_DESC
1210 4966
1210 4967      ; we now have the time string, send it to convert time via
1210 4968      ; an action routine
1210 4969
1210 4970      $STATE
1210 4971      $STRAN TPAS_LAMBDA,TPAS_EXIT,CONVERT_TIME
```

```
1210 4973      ;++
1210 4974      ;
1210 4975      ; subexpressions to get the time string. we will parse out anything
1210 4976      ; up to a blank, or if there is a double quote, we will parse out
1210 4977      ; the entire string with the quotes (including blanks). this is
1210 4978      ; to keep with VMS standard.
1210 4979      ;
1210 4980      ; once we have the time string, we will then send it to an action
1210 4981      ; routine to convert the time into system format. this action routine
1210 4982      ; uses the RTL function CVT_TIME also according to VMS standard.
1210 4983      ;
1210 4984      ;--
1210 4985      ;
1210 4986      $STATE GET_TIME_STRING
1210 4987      $STRAN '""',QUOTED_TIME_STRING
1210 4988      $STRAN !NON_DELIMITER
1210 4989      $STRAN TPAS_LAMBDA,TPAS_EXIT
1210 4990      ;
1210 4991      $STATE
1210 4992      $STRAN TPAS_ANY,GET_TIME_STRING
1210 4993      ;
1210 4994      ;
1210 4995      ; subexpression to parse a quoted time string
1210 4996      ;
1210 4997      ;
1210 4998      $STATE QUOTED_TIME_STRING
1210 4999      $STRAN '""',TPAS_EXIT
1210 5000      $STRAN TPAS_ANY,QUOTED_TIME_STRING
1210 5001      ;
1210 5002      $END_STATE
```

```
1210 5004 ; here are the action routines for the various transitions.
1210 5005
1210 5006
1210 5007 EXPLICIT BLANKS:
0000 1210 5008 :WORD ^M<>
1212 5009 :++
1212 5010 :
1212 5011 : action routine to set TPARSE to accept blanks
1212 5012 :
1212 5013 :
1212 5014 :--
1212 5015
04 AC 01 C8 1212 5016 BISL2 #TPASM_BLANKS,TPASL_OPTIONS(AP) ; turn on blanks
1216 5017
04 1216 5018 RET
1217 5019
1217 5020
1217 5021 IMPLICIT BLANKS:
0000 1217 5022 :WORD ^M<>
1219 5023 :++
1219 5024 :
1219 5025 : action routine to set TPARSE to ignore blanks (its default)
1219 5026 :
1219 5027 :
1219 5028 :--
1219 5029
04 AC 01 CA 1219 5030 BICL2 #TPASM_BLANKS,TPASL_OPTIONS(AP) ; turn off blanks
121D 5031
04 121D 5032 RET
```

```
000C 121E 5034 PRS_FILE_SPEC:
      121E 5035 :WORD ^M<R2,R3>
      1220 5036 :++
      1220 5037 :
      1220 5038 : action routine to use RMS to $PARSE the filespec (pointed to by
      1220 5039 : the token pointer) and to fill in the descriptors of the various
      1220 5040 : fields
      1220 5041 :
      1220 5042 :--
      1220 5043 :
      1220 5044 :
      1220 5045 : adjust the FAB to point to this filespec
      1220 5046 :
      1220 5047 $FAB_STORE FAB = PRSE_FILESPC_FAB,-
      1220 5048 FNA = @<SPEC_DESC+DSC$A_POINTER>,-
      1220 5049 FNS = SPEC_DESC+DSC$W_LENGTH
      1231 5050 :
      1231 5051 $PARSE FAB = PRSE_FILESPC_FAB ; now, parse the filespec
      123C 5052 :
      03 50 E8 123C 5053 BLBS PO,COPY_FILESPEC ; OK, fill in data area
      008D 31 123F 5054 :
      123F 5055 BRW END_PRS_FILE ; if error, then exit with status
      1242 5056 :
      1242 5057 COPY_FILESPEC:
      1242 5058 :
      1242 5059 : the parse was successful, now copy everything into the descriptors
      1242 5060 : RMS $PARSE has put the resultant string in FILE_SPEC_STR
      1242 5061 :
      52 0A30'CF DE 1242 5062 MOVAL PRSE_FILESPC_NAM,R2 ; get the NAM block
      0B A2 9B 1247 5063 :
      077D'CF 124A 5064 MOVZBW NAM$B_ESL(R2),- ; put the size in the descriptor
      40 A2 D0 124D 5065 FILE_SPEC_DESC+DSC$W_LENGTH
      0888'CF 1250 5066 :
      38 A2 9B 1253 5067 MOVL NAM$L_NODE(R2),- ; fill in the node descriptor
      0884'CF 1256 5068 NODE_DESC+DSC$A_POINTER ; ... location
      1259 5069 MOVZBW NAM$B_NODE(R2),- ; ... and size
      44 A2 D0 1259 5070 NODE_DESC+DSC$W_LENGTH
      0890'CF 125C 5071 :
      39 A2 9B 125F 5072 MOVL NAM$L_DEV(R2),- ; fill in the device descriptor
      088C'CF 1262 5073 DEVICE_DESC+DSC$A_POINTER ; ... location
      1265 5074 MOVZBW NAM$B_DEV(R2),- ; ... and size
      48 A2 D0 1265 5075 DEVICE_DESC+DSC$W_LENGTH
      0898'CF 1268 5076 :
      3A A2 9B 1268 5077 MOVL NAM$L_DIR(R2),- ; fill in the directory descriptor
      0894'CF 126E 5078 DIRECTORY_DESC+DSC$A_POINTER ; ... location
      1271 5079 MOVZBW NAM$B_DIR(R2),- ; ... and size
      4C A2 D0 1271 5080 DIRECTORY_DESC+DSC$W_LENGTH
      08A0'CF 1274 5081 :
      3B A2 9B 1277 5082 MOVL NAM$L_NAME(R2),- ; fill in the filename descriptor
      089C'CF 127A 5083 FILENAME_DESC+DSC$A_POINTER ; ... location
      127D 5084 MOVZBW NAM$B_NAME(R2),- ; ... and size
      50 A2 D0 127D 5085 FILENAME_DESC+DSC$W_LENGTH
      08A8'CF 1280 5086 :
      3C A2 9B 1283 5087 MOVL NAM$L_TYPE(R2),- ; fill in the type descriptor
      08A4'CF 1286 5088 TYPE_DESC+DSC$A_POINTER ; ... location
      5090 5089 MOVZBW NAM$B_TYPE(R2),- ; ... and size
      5090 5090 TYPE_DESC+DSC$W_LENGTH
```

```

      54 A2      D0 1289 5091
      08B0'CF    9B 1289 5092      MOVL   NAMS_L_VER(R2),-      ; fill in the version descriptor
      3D A2      9B 128C 5093      VERSION_DESC+DSC$A_POINTER ; ... location
      08AC'CF    9B 128F 5094      MOVZBW NAMS_B_VER(R2),-      ; ... and size
      1292 5095      VERSION_DESC+DSC$W_LENGTH
      1295 5096
      1295 5097 DETERMINE_TYPE:
      1295 5098      ; determine what type of file it is
      1295 5099
      1295 5100      ; if file is a log file, skip type ckeck
      1295 5101
      1295 5102
      31 04A7'CF 00 E0 1295 5103      BBS     #ASSIGN$V_LOG,ASSIGN_CASE,PRS_FILE_SUC ; log file, skip type check
      1298 5104
      08A4'CF    29 1298 5105      CMPC3   TYPE_DESC+DSC$W_LENGTH,-      ; check the type of this filespec
      08A8'DF    29 129F 5106      @<TYPE_DESC+DSC$A_POINTER>,-      ; against that for
      0504'CF    29 12A2 5107      EXE_STR      ; an image
      12A5 5108
      08      12 12A5 5109      BNEQ     10$      ; wasn't .EXE
      12A7 5110
      049F'CF    01 C8 12A7 5111      BISL2   #CRTPRC$M_EXE,CREATE_PROC_CASE ; say that we have an image
      001D      31 12AC 5112      BRW      PRS_FILE_SUC      ; exit with success
      12AF 5113
      12AF 5114 10$:
      12AF 5115      ; wasn't an image, is it a command file?
      12AF 5116
      08A4'CF    29 12AF 5117      CMPC3   TYPE_DESC+DSC$W_LENGTH,-      ; check the type of this filespec
      08A8'DF    29 12B3 5118      @<TYPE_DESC+DSC$A_POINTER>,-      ; against that for
      0508'CF    29 12B6 5119      COM_STR      ; a command procedure
      12B9 5120
      08      12 12B9 5121      BNEQ     20$      ; wasn't .COM
      12BB 5122
      049F'CF    04 C8 12BB 5123      BISL2   #CRTPRC$M_COM,CREATE_PROC_CASE ; say we have a command procedure
      0009      31 12C0 5124      BRW      PRS_FILE_SUC      ; exit with success
      12C3 5125
      12C3 5126 20$:
      12C3 5127      ; wasn't a command procedure, so we have a bad file type
      12C3 5128
      50 00000000'8F D0 12C3 5129      MOVL   #RMSS$ TYP,R0      ; not a valid type
      03      11 12CA 5130      BRB      END_PRS_FILE      ; exit with error
      12CC 5131
      12CC 5132 PRS_FILE_SUC:
      12CC 5133
      12CC 5134      ; all done, so return success status and leave
      12CC 5135
      50 01      D0 12CC 5136      MOVL   #SS$ _NORMAL,R0      ; return success
      12CF 5137
      12CF 5138 END_PRS_FILE:
      12CF 5139
      04 12CF 5140      RET      ; exit
```

```
0000 12D0 5142 CONVERT_TIME:
      12D0 5143 .WORD ^M<>
      12D2 5144
      12D2 5145 ;++
      12D2 5146 ; this action routine takes the time string just parsed out (and
      12D2 5147 ; whose descriptor should be sitting in TPARSE's parameter block)
      12D2 5148 ; and sends it to the RTL function CVT_TIME. CVT_TIME will
      12D2 5149 ; determine if the syntax is acceptable, and will produce the 64-bit
      12D2 5150 ; binary system time if it is.
      12D2 5151 ;
      12D2 5152 ;--
      12D2 5153
      12D2 5154
      09D5'CF 7F 12D2 5155 PUSHAQ TIME_VALUE ; where the system time goes
      09CD'CF 7F 12D6 5156 PUSHAQ TIME_DESC ; the descriptor of the time string
00000000'GF 02 FB 12DA 5157 CALLS #2,G*LIB$CVT_DTIME ; convert the time
      12E1 5158
      04 12E1 5159 RET ; return with status
```

```
003C 12E2 5161 GET_ASG_VAL:
      12E2 5162 .WORD ^M<R2,R3,R4,R5>
      12E4 5163
      12E4 5164
      12E4 5165
      12E4 5166
      12E4 5167
      12E4 5168
      12E4 5169
      12E4 5170
      12E4 5171
      12E4 5172
      12E4 5173
      12E7 5174
      12EA 5175
      12EA 5176
      12ED 5177
      12EF 5178
      12F2 5179
      12F2 5180
      12F5 5181
      12F5 5182
      12F6 5183

      ;++
      ; this action routine is used to copy a "generic" assignment value into
      ; the proper parse buffer. in order for an assignment to be considered
      ; generic, it must be a simple character string which needs no further
      ; parsing.
      ;--

      MOVW TPASL_TOKENCNT(AP),- ; put the size in the descriptor
            ASSIGN_VALUE_DESC+DSCSW_LENGTH ; field

      MOVCL TPASL_TOKENCNT(AP),- ; and move the string there
            @TPASL_TOKENPTR(AP),-
            ASSIGN_VALUE_STR

      MOVL #SSS_NORMAL,R0 ; return success

      RET ; and exit
```

```
003C 12F6 5185 COPY_PARAM:
      12F6 5186 .WORD ^M<R2,R3,R4,R5>
      12F8 5187
      12F8 5188 :++
      12F8 5189 : this action routine is used to copy the parameters associated with
      12F8 5190 : the current filespec into a buffer. it also determines what type
      12F8 5191 : of process the filespec is and sets the proper flag to signal this.
      12F8 5192 :
      12F8 5193 :--
      12F8 5194
      12F8 5195 ; well, we have parameters so first find out what type of file
      12F8 5196 ; we have and set the proper flag saying that that file has parameters
      12F8 5197
      12F8 5198 ; check for valid file types
      12F8 5199
      12F8 5200
      12F8 5201 BBSC #CRTPRC$V_EXE,CREATE_PROC_CASE,EXE_P ; is it an image?
      12FE 5202 BBSC #CRTPRC$V_COM,CREATE_PROC_CASE,COM_P ; is it a command procedure?
      1304 5203
      1304 5204 ; it is not a file type which allows parameters, so exit with error
      1304 5205
      50 5206 CLRL R0 ; signal an error
      0021 31 1306 5207 BRW END_COPY_PARAM ; and leave
      1309 5208
      1309 5209 EXE_P:
      1309 5210 ; it is an image with parameters, set that flag
      1309 5211
      049F'CF 02 C8 1309 5212 BISL2 #CRTPRC$M_EXE_PARM,CREATE_PROC_CASE ; set image with parameters
      0008 31 130E 5213 BRW CONT_COPY ; and copy them
      1311 5214
      1311 5215 COM_P:
      1311 5216 ; it is a command procedure with parameters, set that flag
      1311 5217
      049F'CF 08 C8 1311 5218 BISL2 #CRTPRC$M_COM_PARM,CREATE_PROC_CASE ; set com. with parameters
      0000 31 1316 5219 BRW CONT_COPY ; and copy them
      1319 5220
      1319 5221 CONT_COPY:
      1319 5222 ; now, copy the parameters (the current token) into our parameter
      1319 5223 ; buffer
      1319 5224
      10 AC 28 1319 5225 MOV C3 TPASL_TOKENCNT(AP),- ; move the current token (the
      14 BC 131C 5226 @TPASL_TOKENPTR(AP),- ; parameter list) into our
      0948'CF 131E 5227 PARAM_STR ; parameter buffer
      1321 5228
      10 AC B0 1321 5229 MOVW TPASL_TOKENCNT(AP),- ; and put the size in
      0940'CF 1324 5230 PARAM_DESC+DSC$W_LENGTH ; the descriptor
      1327 5231
      1327 5232 ; exit with success
      1327 5233
      50 01 D0 1327 5234 MOVL #SS$_NORMAL,R0 ; leave with success
      132A 5235
      132A 5236 END_COPY_PARAM:
      132A 5237
      04 132A 5238 RET ; exit
```

			132B	5240	GET_COMMENT:		
		003C	132B	5241	.WORD	^M<R2,R3,R4,R5>	
			132D	5242			
			132D	5243			
			132D	5244			
			132D	5245			
			132D	5246			
			132D	5247			
			132D	5248			
			132D	5249			
			132D	5250			
			132D	5251			
			132D	5252			
08 AC	28		1330	5253	MOV C3	TPASL STRINGCNT(AP),-	; count of remaining characters
0C BC			1332	5254		@TPASL STRINGPTR(AP),-	; location of remaining characters
06F1'CF			1335	5255		COMMENT_STR	; our comment buffer
			1335	5256			
08 AC	B0		1338	5257	MOV W	TPASL STRINGCNT(AP),-	; put the size in our
06E9'CF			133B	5258		COMMENT_DESC+DSC\$W_LENGTH	; comment descriptor
			133B	5259			
50 01	D0		133E	5260	MOV L	#SS\$ _NORMAL,R0	; exit with success
			133E	5261	RET		; and exit

```
0000 133F 5263 TST_DELIM:
      133F 5264 .WORD ^M<>
      1341 5265
      1341 5266 :++
      1341 5267 : action routine to determine if the current token we are looking
      1341 5268 : at is the ending delimiter of a string.
      1341 5269 :
      1341 5270 :--
      1341 5271
      1341 5272
09CC'CF 18 AC 91 1341 5273 CMPB TPASB_CHAR(AP),DELIMIT ; is this character the delimiter
      02 12 1347 5274 BNEQ 10$ ; no, so exit with success
      50 D4 1349 5275 CLRL R0 ; it is, so reject the transition
      1348 5276
      1348 5277 10$:
      04 1348 5278 RET ; leave
      134C 5279
```

```
00000004 134C 5281 .SBTTL Initialize data structure
00000008 134C 5282
0000000C 134C 5283 ; set up offsets for the parameters
00000010 134C 5284
134C 5285 IDS_NUMBER_OF_SLOTS = 4
134C 5286 IDS_SIZE_OF_SLOT = 8
134C 5287 IDS_BEGIN_OF_STRUCT = 12
134C 5288 IDS_LIST_HEADER = 16
134C 5289
134C 5290
134C 5291 :++
134C 5292 : FUNCTIONAL DESCRIPTION:
134C 5293 :
134C 5294 : This subroutine is called to initialize the process information
134C 5295 : data structure. The data structure is to be a linked list of
134C 5296 : individual "slots" where one slot is allocated per user. This
134C 5297 : routine sets up an available reservoir of free slots. The header
134C 5298 : points to the first slot and then the linked list continues from there.
134C 5299 :
134C 5300 : NOTE: it is a requirement that structure area be a contiguous area.
134C 5301 : (There may, however, be separate non-contiguous structure areas.)
134C 5302 :
134C 5303 : CALLING SEQUENCE:
134C 5304 :
134C 5305 : This routine is called by
134C 5306 : PUSHAQ LIST_HEADER
134C 5307 : PUSHAL BEGIN_OF_STRUCT
134C 5308 : PUSHL SIZE_OF_SLOT
134C 5309 : PUSHL NUMBER_OF_SLOTS
134C 5310 : CALLS #4, INIT_STRUCT
134C 5311 :
134C 5312 : INPUT PARAMETERS:
134C 5313 :
134C 5314 : IDS_NUMBER_OF_SLOTS(AP) -- this is a value of the total number of slots to
134C 5315 : be in the structure.
134C 5316 : IDS_SIZE_OF_SLOT(AP) -- this is a value of the size of each individual
134C 5317 : slot
134C 5318 : IDS_BEGIN_OF_STRUCT(AP) -- this is the address of the beginning of the
134C 5319 : entire structure.
134C 5320 : IDS_LIST_HEADER(AP) -- this is the address of a quadword which will be the
134C 5321 : available list header for the structure (hold FLINK and BLINK)
134C 5322 :
134C 5323 : IMPLICIT INPUTS:
134C 5324 :
134C 5325 : None.
134C 5326 :
134C 5327 : OUTPUT PARAMETERS:
134C 5328 :
134C 5329 : None.
134C 5330 :
134C 5331 : IMPLICIT OUTPUTS:
134C 5332 :
134C 5333 : None.
134C 5334 :
134C 5335 : COMPLETION CODES:
134C 5336 :
134C 5337 : SSS_NORMAL
```

```
134C 5338 :  
134C 5339 : SIDE EFFECTS:  
134C 5340 :  
134C 5341 : Links throughout the structure will be modified.  
134C 5342 :--  
134C 5343 :  
003C 134C 5344 INIT_STRUCT:  
134C 5345 .WORD ^M<R2,R3,R4,R5>  
134E 5346 :  
134E 5347 ; move the parameters into registers  
134E 5348 :  
52 04 AC D0 134E 5349 MOVL IDS_NUMBER OF SLOTS(AP),R2 ; number of slots to init  
53 08 AC D0 1352 5350 MOVL IDS_SIZE OF SCOT(AP),R3 ; size of each slot  
54 0C AC D0 1356 5351 MOVL IDS_BEGIN OF STRUCT(AP),R4 ; addr of top of structure  
55 10 AC D0 135A 5352 MOVL IDS_LIST_HEADER(AP),R5 ; addr of structure header  
135E 5353 :  
135E 5354 ; now insert the various slots into a linked stack. The slots  
135E 5355 ; are inserted at the tail so that the header will be pointing to  
135E 5356 ; low order memory and will work towards high order.  
135E 5357 :  
65 64 5D 135E 5358 10$:  
54 53 C0 1361 5360 INSQT1 (R4),(R5) ; put slot at tail of list  
F7 52 F5 1364 5361 ADDL2 R3,R4 ; get address of next slot  
1367 5362 SOBGTR R2,10$ ; go through whole list  
50 01 D0 1367 5363 MOVL #SS$ _NORMAL,R0 ; return success  
136A 5364 :  
04 136A 5365 RET
```

```
00000004 136B 5367 .SBTTL Expand region for more data areas
00000008 136B 5368
0000000C 136B 5369 ; set up offsets for the parameters
136B 5370
136B 5371 ER_NUMBER_OF_SLOTS = 4
136B 5372 ER_SIZE_OF_SLOT = 8
136B 5373 ER_LOCATION_OF_STRUCT = 12
136B 5374
136B 5375
136B 5376 :++
136B 5377 : FUNCTIONAL DESCRIPTION:
136B 5378 :
136B 5379 : This routine is called if it is required to expand the current
136B 5380 : virtual address space because the current number of slots in the
136B 5381 : data structure are filled. It does this by first calculating the
136B 5382 : number of pages needed and then by doing an $EXPREG system service.
136B 5383 :
136B 5384 : CALLING SEQUENCE:
136B 5385 :
136B 5386 : This routine is called by
136B 5387 : PUSHAQ LOCATION_OF_STRUCT
136B 5388 : PUSHL SIZE_OF_SLOT
136B 5389 : PUSHL NUMBER_OF_SLOTS
136B 5390 : CALLS #3,EXPND_REGION
136B 5391 :
136B 5392 : INPUT PARAMETERS:
136B 5393 :
136B 5394 : ER_NUMBER_OF_SLOTS(AP) -- this is a value of the total number of slots to
136B 5395 : be in the structure.
136B 5396 : ER_SIZE_OF_SLOT(AP) -- this is a value of the size of each individual slot
136B 5397 : ER_LOCATION_OF_STRUCT(AP) -- this is the address of a quadword to hold the
136B 5398 : starting location (first longword) of the expanded region and
136B 5399 : the ending location (second longword) of the expanded region.
136B 5400 :
136B 5401 : IMPLICIT INPUTS:
136B 5402 :
136B 5403 : None.
136B 5404 :
136B 5405 : OUTPUT PARAMETERS:
136B 5406 :
136B 5407 : None.
136B 5408 :
136B 5409 : IMPLICIT OUTPUTS:
136B 5410 :
136B 5411 : @ER_LOCATION_OF_STRUCT(AP) will hold the starting and the ending location
136B 5412 : of the expanded region.
136B 5413 :
136B 5414 : COMPLETION CODES:
136B 5415 :
136B 5416 : $$$_NORMAL
136B 5417 : $$$_SSFAIL -- failed to expand region, system service failed
136B 5418 : $$$_FLTDIV -- attempted divide by zero (note: FAO arguments are not
136B 5419 : included with this code).
136B 5420 :
136B 5421 : SIDE EFFECTS:
136B 5422 :
136B 5423 : None.
```

```
136B 5424 ;--
136B 5425
136B 5426 EXPND_REGION:
007C 136B 5427 .WORD ^M<R2,R3,R4,R5,R6>
136D 5428
136D 5429
136D 5430 ; move the parameters into registers
136D 5431
136D 5432 MOVL ER_NUMBER_OF_SLOTS(AP),R2 ; number of slots to init
52 04 AC D0 1371 5433 MOVL ER_SIZE_OF_SLOT(AP),R3 ; size of each slot
53 08 AC D0 1375 5434 MOVL ER_LOCATION_OF_STRUCT(AP),R4 ; addr of top of structure
54 0C AC D0 1379 5435
1379 5436 ; calculate the number of pages we are to expand by
1379 5437
1379 5438 ; first see if we are going to divide by zero
1379 5439
1379 5440 TSTL #PAGE_SIZE ; make sure page size isn't zero
137F 5441 BNEQ 10$ ; isn't zero, so continue
50 00000494 8F D0 1381 5442 MOVL #SS$_FLTDIV,R0 ; tried div-by-zero, so error
004F 31 1388 5443 BRW END_EXPND_REGION ; and leave
138B 5444
138B 5445 ; OK, now calculate the number of pages
138B 5446
138B 5447 10$:
55 55 53 52 C5 138B 5448 MULL3 R2,R3,R5 ; total number of bytes needed
55 000001FF 8F C0 138F 5449 ADDL2 #<PAGE_SIZE-1>,R5 ; add another page minus 1 to
; do the rounding up
55 00000200 8F C6 1396 5451 DIVL2 #PAGE_SIZE,R5 ; and get the number of pages
139D 5452
139D 5453 ; we now know how many pages we need, so do the expansion
139D 5454
139D 5455 $SETSFM_S ENBFLG = #0 ; errors are OK, turn failure off
09 56 D4 13A6 5456 CLRL R6 ; assume failure was disabled
50 D1 13A8 5457 CMPL R0,#SS$_WASSET ; was SS failure mode disabled?
03 12 13AB 5458 BNEQ 20$ ; no, so continue
56 01 D0 13AD 5459 MOVL #1,R6 ; save it to be re-enabled
13B0 5460
13B0 5461 20$:
13B0 5462 $EXPREG_S PAGCNT = R5,- ; expand by this number of pages
13B0 5463 RETADR = (R4) ; put results here
13BF 5464
13BF 5465 MOVL #SS$_NORMAL,R2 ; assume it succeeded
13C2 5466
13C2 5467 CMPL R0,#SS$_NORMAL ; did it succeed?
01 50 D1 13C5 5468 BEQL 30$ ; yes, so exit
07 13 13C7 5469 MOVL #SS$_SSFAIL,R2 ; no, so return error
52 0000045C 8F D0 13CE 5470
13CE 5471 30$:
13CE 5472 $SETSFM_S ENBFLG = R6 ; set to previous state
13D7 5473
13D7 5474 MOVL R2,R0 ; return the completion code
50 52 D0 13DA 5475
13DA 5476 END_EXPND_REGION:
04 13DA 5477 RET ; and leave
```

```
13DB 5479 .SBTTL Abort Detached Processes
13DB 5480
13DB 5481 :++
13DB 5482 : FUNCTIONAL DESCRIPTION:
13DB 5483 :
13DB 5484 : This routine is used to stop all the detached processes created by
13DB 5485 : this TSTCNTRL. It does this by going down the current process list
13DB 5486 : and doing a $FORCEX on each of the processes. It will also set a bit
13DB 5487 : (PRCINFO$V_PROC_ABORTED) in each of the processes information block.
13DB 5488 : This bit is to signal the termination routine that this process was
13DB 5489 : aborted prematurely. It then sets a timer to give the processes a
13DB 5490 : chance to run down before they are nuked.
13DB 5491 :
13DB 5492 : CALLING SEQUENCE:
13DB 5493 :
13DB 5494 : CALLS #0,ABORT_PROCESSES
13DB 5495 :
13DB 5496 : INPUT PARAMETERS:
13DB 5497 :
13DB 5498 : None.
13DB 5499 :
13DB 5500 : IMPLICIT INPUTS:
13DB 5501 :
13DB 5502 : FLAGS -- the TSTCNTRL flags
13DB 5503 : CURRENT_LIST_HEAD -- location of the header for the list of currently
13DB 5504 : running processes.
13DB 5505 :
13DB 5506 : OUTPUT PARAMETERS:
13DB 5507 :
13DB 5508 : None.
13DB 5509 :
13DB 5510 : IMPLICIT OUTPUTS:
13DB 5511 :
13DB 5512 : Sets a bit (PRCINFO$V_PROC_ABORTED) in each processes information block.
13DB 5513 :
13DB 5514 : COMPLETION CODES:
13DB 5515 :
13DB 5516 : $$$_NORMAL
13DB 5517 :
13DB 5518 : SIDE EFFECTS:
13DB 5519 :
13DB 5520 : Causes the current detached processes to run down and leave the system.
13DB 5521 : --
13DB 5522 :
13DB 5523 : ABORT_PROCESSES:
001C 13DB 5524 : .WORD ^M<R2,R3,R4>
13DB 5525 :
13DB 5526 : ; get the list of currently running processes
13DB 5527 :
53 4660'CF 7E 13DB 5528 MOVAQ CURRENT_LIST_HEAD,R3 ; header of currently running
13DB 5529 ; processes
63 05 13E2 5530 TSTL (R3) ; is the list empty?
13E4 5531
03 12 13E4 5532 BNEU 108 ; no, so stop current processes
13E6 5533
003E 31 13E6 5534 BRW END_ABORT_PROC ; no current processes, so exit
13E9 5535
```

```

13E9 5536 10$:
13E9 5537 ; go through list and stop all the processes
13E9 5538
54 52 63 D0 13E9 5539 MOVL PRCINFO$_FLINK(R3),R2 ; get offset to next slot
53 52 C1 13EC 5540 ADDL3 R2,R3,R4 ; calculate address of next slot
13F0 5541
13F0 5542 STOP_PROC:
13F0 5543
52 64 D0 13F0 5544 MOVL PRCINFO$_FLINK(R4),R2 ; get offset to next slot
13F3 5545
13F3 5546 ; stop the next process on the list . . .
13F3 5547
13F3 5548 $SETSFMS ENBFLG = #0 ; check this error by hand
13FC 5549
13FC 5550 $FORCEX$_PIDADR = PRCINFO$_PID(R4) ; abort the process
140A 5551
140A 5552 ; see if process was aborted without error. if it had error, it may
140A 5553 ; be that the process was already terminated; so don't touch abort
140A 5554 ; bit
140A 5555
50 01 D1 140A 5556 CMPL #SS$_NORMAL,R0 ; did it abort OK?
04 12 140D 5557 BNEQ 10$ ; no, so don't set abort bit
02 C8 140F 5558
38 A4 140F 5559 BISL2 #PRCINFO$_PROC_ABORTED,- ; set bit saying this process
1411 5560 PRCINFO$_FLAG$R4) ; was aborted
1413 5561
1413 5562 10$:
1413 5563 $SETSFMS ENBFLG = #1 ; turn on failure mode
141C 5564
141C 5565 ; . . . stopped that process, now see if there is another one
141C 5566
54 52 C0 141C 5567 ADDL2 R2,R4 ; calculate address of next slot
53 54 D1 141F 5568 CMPL R4,R3 ; are we at the end of the list?
03 13 1422 5570 BEQL END_ABORT_PROC ; yes, so leave
FFC9 31 1424 5572 BRW STOP_PROC ; more processes left
1427 5574
1427 5575 END_ABORT_PROC:
1427 5576 ; now set the timer to insure that at least one process terminates in
1427 5577 ; a specified amount of time. If the timer goes off (ie, no process
1427 5578 ; terminated), then all of the processes are to be nuked
1427 5579
1427 5580 $SETIMP$_DAYTIM = PROC_TERM_DELT,- ; set watchdog timer to insure
1427 5581 ASTADR = NUKE_PROC_AST,- ; that processes start to
1427 5582 REQIDT = #ABRT_TIM_ID ; terminate
143A 5583
50 01 D0 143A 5584 MOVL #SS$_NORMAL,R0 ; return success
04 143D 5585 RET
143D 5586
```

```
143E 5588 .SBTTL Nuke Detached Processes AST
143E 5589
143E 5590 :++
143E 5591 : FUNCTIONAL DESCRIPTION:
143E 5592 :
143E 5593 : This routine is called as a last resort when detached processes
143E 5594 : previously aborted by $FORCEX have still failed to terminate. It
143E 5595 : is called when not one process has terminated within a set waiting
143E 5596 : time (determined by ABRT_DELT_SCNDS). This routine will go through
143E 5597 : the current process list and do a $DELPRC on each of the processes
143E 5598 : still in that list.
143E 5599 :
143E 5600 : CALLING SEQUENCE:
143E 5601 :
143E 5602 : Called by AST when abort process timer expires.
143E 5603 :
143E 5604 : INPUT PARAMETERS:
143E 5605 :
143E 5606 : None.
143E 5607 :
143E 5608 : IMPLICIT INPUTS:
143E 5609 :
143E 5610 : FLAGS -- the TSTCNTRL flags
143E 5611 : CURRENT_LIST_HEAD -- location of the header for the list of currently
143E 5612 : running processes.
143E 5613 :
143E 5614 : OUTPUT PARAMETERS:
143E 5615 :
143E 5616 : None.
143E 5617 :
143E 5618 : IMPLICIT OUTPUTS:
143E 5619 :
143E 5620 : None.
143E 5621 :
143E 5622 : COMPLETION CODES:
143E 5623 :
143E 5624 : SS$_NORMAL
143E 5625 :
143E 5626 : SIDE EFFECTS:
143E 5627 :
143E 5628 : Aborts detached process in the middle of whatever it is doing.
143E 5629 : May cause detached process not to clean up when it leaves.
143E 5630 :--
143E 5631 :
143E 5632 : NUKE_PROC AST:
001C 143E 5633 .WORD ^M<R2,R3,R4>
1440 5634
1440 5635 ; get the list of currently running processes
1440 5636
53 4660'CF 7E 1440 5637 MCVAQ CURRENT_LIST_HEAD,R3 ; header of currently running
1445 5638 ; processes
63 D5 1445 5639 TSTL (R3) ; is the list empty?
1447 5640
03 12 1447 5641 BNEQ 10$ ; no, so stop current processes
1449 5642
0038 31 1449 5643 BRW END_NUKE_PROC ; no current processes, so exit
144C 5644
```

```
144C 5645 10$:  
144C 5646 ; go through list and stop all the processes  
144C 5647  
54 52 63 D0 144C 5648 MOVL PRCINFO$A_FLINK(R3),R2 ; get offset to next slot  
53 52 C1 144F 5649 ADDL3 R2,R3,R4 ; calculate address of next slot  
1453 5650  
1453 5651 NUKE_PROC:  
1453 5652  
1453 5653 ; stop the next process on the list . . .  
1453 5654  
1453 5655 $SETSFM_S ENBFLG = #0 ; check this error by hand  
145C 5656  
145C 5657 $DELPRC_S PIDADR = PRCINFO$L_PID(R4) ; abort the process  
1468 5658  
1468 5659 ; see if process was aborted without error. if it had error, it may  
1468 5660 ; be that the process was already terminated  
1468 5661  
50 01 D1 1468 5662 CMPL #SS$_NORMAL,R0 ; did it abort OK?  
00 12 1468 5663 BNEQ 10$ ; no, so don't set abort bit  
146D 5664  
146D 5665 10$:  
146D 5666 $SETSFM_S ENBFLG = #1 ; turn on failure mode  
1476 5667  
1476 5668 ; . . . stopped that process, now see if there is another one  
1476 5669  
52 64 D0 1476 5670 MOVL PRCINFO$A_FLINK(R4),R2 ; get offset to next slot  
54 52 C0 1479 5671 ADDL2 R2,R4 ; calculate address of next slot  
147C 5672  
53 54 D1 147C 5673 CMPL R4,R3 ; are we at the end of the list?  
147F 5674  
03 13 147F 5675 BEQL END_NUKE_PROC ; yes, so leave  
1481 5676  
FFCF 31 1481 5677 BRW NUKE_PROC ; more processes left  
1484 5678  
1484 5679 END_NUKE_PROC:  
1484 5680  
04A3'CF 00010000 8F C8 1484 5681 BISL2 #TCNTRL$M_PROC_NUKED,FLAGS ; set flag saying processes have  
148D 5682 ; been nuked  
148D 5683  
50 01 D0 148D 5684 MOVL #SS$_NORMAL,R0 ; return success  
1490 5685  
04 1490 5686 RET  
1491 5687
```

```
1491 5689 .SBTTL Process Termination AST
1491 5690 :++
1491 5691 : FUNCTIONAL DESCRIPTION:
1491 5692 :
1491 5693 : This routine is called when a currently executing detached process
1491 5694 : terminates. The terminated process sends a mailbox message back to the
1491 5695 : the TSTCNTRL and notification of that mailbox sends control here.
1491 5696 : This routine then locates the process info slot of the terminated
1491 5697 : process, removes it from the current process list, fills it in with
1491 5698 : process termination info, places it on the completed process list,
1491 5699 : and signals that a process has terminated.
1491 5700 :
1491 5701 :
1491 5702 : CALLING SEQUENCE:
1491 5703 :
1491 5704 : This is called as an AST routine when a termination mailbox is
1491 5705 : received.
1491 5706 :
1491 5707 : INPUT PARAMETERS:
1491 5708 :
1491 5709 : None
1491 5710 :
1491 5711 : IMPLICIT INPUTS:
1491 5712 :
1491 5713 : The termination mailbox.
1491 5714 : CURRENT_LIST_HEAD - header pointing to the list of currently running
1491 5715 : processes.
1491 5716 : COMPLETE_Q_HEAD - header pointing to the queue of completed processes.
1491 5717 : PROCESS_TERM_IOSVB - the IOSB for the termination mailbox
1491 5718 :
1491 5719 : OUTPUT PARAMETERS:
1491 5720 :
1491 5721 : None
1491 5722 :
1491 5723 : IMPLICIT OUTPUTS:
1491 5724 :
1491 5725 : CONTROL_CASE - the set of flags to determine which case statement to
1491 5726 : perform next.
1491 5727 :
1491 5728 : COMPLETION CODES:
1491 5729 :
1491 5730 : None.
1491 5731 :
1491 5732 : SIDE EFFECTS:
1491 5733 :
1491 5734 : Will modify both the current process list and the complete process queue.
1491 5735 : Will change the control of the main case loop by setting the
1491 5736 : CASE$V_PROC_TERM flag.
1491 5737 : If unable to find process info slot, it will not set CASE$V_PROC_TERM.
1491 5738 :--
1491 5739 :
1491 5740 :
1491 5741 : PROC_TERM_AST:
1491 5742 : .WORD ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
1493 5743 :
1493 5744 : MOVAL SSERROR,(FP) ; declare exception handler for
1498 5745 : ; this routine
```

				1498	5746		
				1498	5747		
				1498	5748		; see if the QIO to the mailbox has been canceled or aborted
				1498	5749		; the QIO may have been canceled if it came during the exit handler,
				1498	5750		; if this is the case, ignore the QIO
04AB'CF	2C	B1		1498	5751	CMPW	#SS\$_ABORT,PROC_TERM_IOSB ; has QIO been aborted?
	03	12		149D	5752	BNEQ	10\$; no, see if canceled
	00A8	31		149F	5753	BRW	END_PROC_TERM_AST ; yes, exit
				14A2	5754		
				14A2	5755	10\$:	
04AB'CF	0830	8F	B1	14A2	5756	CMPW	#SS\$_CANCEL,PROC_TERM_IOSB ; has QIO been canceled?
	03	12		14A9	5757	BNEQ	20\$; no, so continue
	009C	31		14AB	5758	BRW	END_PROC_TERM_AST ; yes, exit
				14AE	5759		
				14AE	5760	20\$:	
				14AE	5761		; now, find the process info slot for this particular process
				14AE	5762		
5B	011F'CF	DE		14AE	5763	MOVAL	TERM_BUFF,R11 ; get address of termination
				14B3	5764		; mailbox
				14B3	5765		
	08 AB	DD		14B3	5766	PUSHL	ACCSL_PID(R11) ; find slot for this PID
	4660'CF	7F		14B6	5767	PUSHAQ	CURRENT_LIST_HEAD ; look for it in this list
1559'CF	02	FB		14BA	5768	CALLS	#2,FIND_PROC_SLOT ; and find the slot
				14BF	5769		
				14BF	5770		; see if we were able to find the slot
				14BF	5771		
50	01	D1		14BF	5772	CMPL	#SS\$_NORMAL,R0 ; did it find the slot
	03	13		14C2	5773	BEQL	30\$; yes, so continue
				14C4	5774		
	0083	31		14C4	5775	BRW	END_PROC_TERM_AST ; no, so leave without setting flag
				14C7	5776		
				14C7	5777	30\$:	
				14C7	5778		; we now have the address of the slot, the status was returned in
				14C7	5779		; R0 and the address of the slot in R1
				14C7	5780		
5A	51	D0		14C7	5781	MOVL	R1,R10 ; save address of slot
				14CA	5782		
				14CA	5783		; now, remove the slot from the current list and put it at the
				14CA	5784		; tail of the completed queue. we must remove the slot by the
				14CA	5785		; following equations: BLINK(FLINK(SLOT)) = BLINK(SLOT) and
				14CA	5786		; FLINK(BLINK(SLOT)) = FLINK(SLOT).
				14CA	5787		; since the values in the FLINK's and BLINK's are offsets, we must
				14CA	5788		; calculate the actual addresses and then calculate the new offsets
				14CA	5789		
53	52	6A	D0	14CA	5790	MOVL	PRCINFO\$_FLINK(R10),R2 ; get offset to forward struct
	52	5A	C1	14CD	5791	ADDL3	R10,R2,R3 ; get addr. of forward structure
				14D1	5792		
54	04	AA	D0	14D1	5793	MCVL	PRCINFO\$_BLINK(R10),R4 ; get offset to backward struct
55	54	5A	C1	14D5	5794	ADDL3	R10,R4,R5 ; get addr. of backward structure
				14D9	5795		
				14D9	5796		; now calculate new offsets and change links
				14D9	5797		
56	55	53	C3	14D9	5798	SUBL3	R3,R5,R6 ; calc new offset of back. struct
	04	A3	D0	14DD	5799	MOVL	R6,PRCINFO\$_BLINK(R3) ; put offset of backward struct
				14E1	5800		; in BLINK of forward structure
				14E1	5801		
56	53	55	C3	14E1	5802	SUBL3	R5,R3,R6 ; calc new offset of for. struct

```

        65    56    D0    14E5    5803    MOVL    R6,PRCINFO$A_FLINK(R5)          ; put offset of forward struct
        14E8    5804                                ; in FLINK of backward structure
        14E8    5805
        14E8    5806    ; now put the slot on the tail of the completed queue
        14E8    5807
        4658'CF    6A    5D    14E8    5808    INSQTI (R10),COMPLETE_Q_HEAD      ; put slot at end of complete queue
        14ED    5809
        14ED    5810    ; now move items from the termination mailbox into the process info slot
        14ED    5811
        08 AA    04 AB    D0    14ED    5812    MOVL    ACC$$_FINALSTS(R11),PRCINFO$_FINALSTS(R10) ; the final status
        14F2    5813
        10 AA    10 AB    7D    14F2    5814    MOVQ    ACC$_Q_TERMTIME(R11),PRCINFO$_Q_TERMTIME(R10) ; the termination time
        14F7    5815
        18 AA    2C AB    D0    14F7    5816    MOVL    ACC$_CPUTIM(R11),PRCINFO$_CPUTIM(R10) ; CPU time used
        14FC    5817
        1C AA    30 AB    D0    14FC    5818    MOVL    ACC$_PAGEFLTS(R11),PRCINFO$_PAGEFLTS(R10) ; page faults incurred
        1501    5819
        20 AA    34 AB    D0    1501    5820    MOVL    ACC$_PGFLPEAK(R11),PRCINFO$_PGFLPEAK(R10) ; peak page file usage
        1506    5821
        24 AA    38 AB    D0    1506    5822    MOVL    ACC$_WSPEAK(R11),PRCINFO$_WSPEAK(R10) ; working set peak
        150B    5823
        28 AA    3C AB    D0    150B    5824    MOVL    ACC$_BIOCNT(R11),PRCINFO$_BIOCNT(R10) ; buffered I/O operations
        1510    5825
        2C AA    40 AB    D0    1510    5826    MOVL    ACC$_DIOCNT(R11),PRCINFO$_DIOCNT(R10) ; direct I/O operations
        1515    5827
        30 AA    48 AB    7D    1515    5828    MOVQ    ACC$_Q_LOGIN(R11),PRCINFO$_Q_LOGIN(R10) ; time process logged in
        151A    5829
        151A    5830    ; we are all done here, signal for control to go to the process
        151A    5831    ; termination routine
        151A    5832
        049B'CF    04    C8    151A    5833    BISL2  #CASE$_PROC_TERM,CONTROL_CASE ; handle the terminated process
        151F    5834
        151F    5835    ; now activate another read to get the next termination mailbox
        151F    5836
        151F    5837    $QIO_S  CHAN    = MBX_CHAN,-          ; try & read another mailbox
        151F    5838        FUNC    = #IOS_READVBLK,-
        151F    5839        ASTADR  = PROC_TERM_AST,-
        151F    5840        IOSB    = PROC_TERM_IOSB,-
        151F    5841        P1      = TERM_BUFF,-
        151F    5842        P2      = #MAILBOX_SIZE
        154A    5843
        154A    5844    END_PROC_TERM_AST:
        154A    5845
        154A    5846    $WAKE_S          ; wake up and service this
        1555    5847        ; terminated process
        1555    5848
        50    01    D0    1555    5849    MOVL    #$$$_NORMAL,R0          ; return success
        1558    5850
        04    1558    5851    RET          ; return to where we were
```

```
1559 5853 .SBTTL Find Process Slot
1559 5854
1559 5855 ; define offsets for parameters
1559 5856
00000004 1559 5857 FS_LIST_HEADER = 4
00000008 1559 5858 FS_SLOT_PID = 8
1559 5859
1559 5860 ;++
1559 5861 : FUNCTIONAL DESCRIPTION:
1559 5862 :
1559 5863 : This routine is called from the AST routine PROC_TERM_AST and it is
1559 5864 : used to go sequentially through the Current Process List to find the
1559 5865 : the slot which corresponds to the process which just terminated. It
1559 5866 : does this by checking the PID's from the Current Process List with that
1559 5867 : of the terminated process until it finds a match. It then returns the
1559 5868 : address of the slot in R1.
1559 5869 :
1559 5870 : CALLING SEQUENCE:
1559 5871 :
1559 5872 :     PUSHL  PID
1559 5873 :     PUSHAQ CURRENT_LIST_HEAD
1559 5874 :     CALLS  #2,FIND_PROC_SLOT
1559 5875 :
1559 5876 : INPUT PARAMETERS:
1559 5877 :
1559 5878 :     FS_LIST_HEADER(AP) - address of the header of the Current Process List
1559 5879 :     FS_SLOT_PID(AP)    - value of the PID of the completed process
1559 5880 :
1559 5881 : IMPLICIT INPUTS:
1559 5882 :
1559 5883 :     None.
1559 5884 :
1559 5885 : OUTPUT PARAMETERS:
1559 5886 :
1559 5887 :     None
1559 5888 :
1559 5889 : IMPLICIT OUTPUTS:
1559 5890 :
1559 5891 :     R1 holds the address of the slot of the just completed process
1559 5892 :
1559 5893 : COMPLETION CODES:
1559 5894 :
1559 5895 :     SSS_NORMAL
1559 5896 :     SSS_NOSUCHSEC - we couldn't find the specified slot in the data structure
1559 5897 :
1559 5898 : SIDE EFFECTS:
1559 5899 :
1559 5900 :     None.
1559 5901 :--
1559 5902
0004 1559 5903 FIND_PROC_SLOT:
1559 5904 .WORD *M<R2>
1559 5905
1559 5906 ; get the address of the current process list, this will be the base
1559 5907
52 04 AC D0 1559 5908 MOVL FS_LIST_HEADER(AP),R2 ; base of the list
1559 5909
```

```

        62  00  D1 155F 5910      CMPL  #0,(R2)          ; is the header empty?
        19  13  1562 5911      BEQL  SLOT_NOT_FND      ; yes, then no slot
        1564 5912
        1564 5913 10$:
        1564 5914      ; now calculate the address of the next slot
        52  62  C0 1564 5916      ADDL2  PRCINFO$A_FLINK(R2),R2      ; this is start of next slot
        1567 5917
        1567 5918      ; see if we are back to the header
        1567 5919
        04 BC  52  D1 1567 5920      CMPL  R2,@FS_LIST_HEADER(AP)      ; are back to the header?
        10  13  1568 5921      BEQL  SLOT_NOT_FND      ; yes, and we didn't find the slot
        156D 5922
        156D 5923      ; we have a valid slot, so check the PID and see if it is the one
        156D 5924      ; we want
        156D 5925
        0C A2  08 AC  D1 156D 5926      CMPL  FS_SLOT_PID(AP),PRCINFO$A_PID(R2) ; is this the right slot?
        FO  12  1572 5927      BNEQ  10$              ; no, so try again
        1574 5928
        1574 5929      ; we found the slot we were looking for, get the address of it
        1574 5930
        51  52  D0 1574 5931      MOVL  R2,R1              ; return the address of the slot
        50  01  D0 1577 5932      MOVL  #SS$NORMAL,R0      ; return success
        001A 31 157A 5933      BRW   END_FIND_PROC_SLOT    ; and leave
        157D 5934
        157D 5935 SLOT_NOT_FND:
        157D 5936      ; we went through the entire data structure and didn't find the right
        157D 5937      ; slot. Something must be very wrong. Write out a message and return
        157D 5938      ; an error status.
        157D 5939
        157D 5940      ; write out an error message
        157D 5941
        01E9'CF  DF 157D 5942      PUSHAL DATSTRUC_ERR      ; an error with the data structure
        01  DD 1581 5943      PUSHL  #1
        00C31130 8F  DD 1583 5944      PUSHL  #TCNTRL$TEXT!STSS$K_WARNING ; we can try and continue
        00000000'GF 03 FB 1589 5945      CALLS  #3,G^LIB$SIGNAL ; write the error
        1590 5946
        1590 5947      ; and return an error status
        1590 5948
        50  00000978 8F  D0 1590 5949      MOVL  #SS$NOSUCHSEC,R0      ; say we couldn't find that slot
        1597 5950
        1597 5951      END_FIND_PROC_SLOT:
        1597 5952
        04 1597 5953      RET              ; and return
```

```
1598 5955 .SBTTL Maximum Timer Expiration AST
1598 5956
1598 5957 :++
1598 5958 : FUNCTIONAL DESCRIPTION:
1598 5959 :
1598 5960 : This routine is called if the maximum time as defined by the
1598 5961 : assignment statement MAXTIME = hh:mm has expired. The routine
1598 5962 : prints out a message saying that max-time has expired, and sets
1598 5963 : the case bit CASE$V_TIME_EXP. Control then returns to the case loop
1598 5964 : where proper action for the termination of currently running processes
1598 5965 : will take place.
1598 5966
1598 5967 : CALLING SEQUENCE:
1598 5968 :
1598 5969 : Called via an AST when timer expires.
1598 5970
1598 5971 : INPUT PARAMETERS:
1598 5972 :
1598 5973 : None.
1598 5974
1598 5975 : IMPLICIT INPUTS:
1598 5976 :
1598 5977 : None.
1598 5978
1598 5979 : OUTPUT PARAMETERS:
1598 5980 :
1598 5981 : None.
1598 5982
1598 5983 : IMPLICIT OUTPUTS:
1598 5984 :
1598 5985 : The CASE$V_TIME_EXP bit will be set in CONTROL_CASE.
1598 5986
1598 5987 : COMPLETION CODES:
1598 5988 :
1598 5989 : SSS_NORMAL
1598 5990
1598 5991 : SIDE EFFECTS:
1598 5992 :
1598 5993 : Will cause the MAX_TIME_EXP routine to be called to stop all
1598 5994 : currently running processes.
1598 5995 :
1598 5996 :--
1598 5997
1598 5998 TIME_EXP_AST:
1598 5999 .WORD ^M<> ; Entry mask
159A 6000
159A 6001 MOVAL SSERROR,(FP) ; declare exception handler for
159F 6002 ; this routine
159F 6003
159F 6004 ; set case bit
159F 6005
159F 6006 BISL2 #CASE$M_TIME_EXP,CONTROL_CASE ; case to proper routine
15A4 6007
15A4 6008 $WAKE_S ; do a wake to get out of possible
15AF 6009 ; HIBER
15AF 6010
15AF 6011 MOVL #SS$NORMAL,R0 ; return success
```

0000

6D 15B3'CF DE

049B'CF 02 C8

50 01 D0

TSTCNTRL
V04-000

TEST PACKAGE CONTROL PROGRAM
Maximum Timer Expiration AST

L 11

```
16-SEP-1984 01:30:05 VAX/VMS Macro V04-00
5-SEP-1984 04:26:03 [UETP.SRC]JETPHAS00.MAR;1
```

Page 139
(70)

```

04 1582 6012
   1582 6013      RET

```

[illegible]

```
1583 6015 .SBTTL System Service Exception Handler
1583 6016 :++
1583 6017 : FUNCTIONAL DESCRIPTION:
1583 6018 :
1583 6019 : This routine is executed if a system service or RMS error occurs or
1583 6020 : if a LIB$SIGNAL system service is used to output a message.
1583 6021 : Information about this method of handling messages and errors can be
1583 6022 : found in the VMS COMMON RUN-TIME manual and in the VMS SYSTEM SERVICE
1583 6023 : manual.
1583 6024 :
1583 6025 : If a severe error or exception occurs, then the TSTCNTRL begins
1583 6026 : the abort process to run down all currently running processes. This
1583 6027 : is done by calling the exit handler. If a severe error or exception
1583 6028 : occurs while in the abort process, then the TSTCNTRL gives up trying
1583 6029 : to run down processes and simply exits.
1583 6030 :
1583 6031 : CALLING SEQUENCE:
1583 6032 :
1583 6033 : Entered via an exception from the system
1583 6034 :
1583 6035 : INPUT PARAMETERS:
1583 6036 :
1583 6037 :
1583 6038 : AP ---->
1583 6039 :
1583 6040 :
1583 6041 :
1583 6042 :
1583 6043 :
1583 6044 :
1583 6045 :
1583 6046 :
1583 6047 :
1583 6048 :
1583 6049 :
1583 6050 :
1583 6051 :
1583 6052 :
1583 6053 :
1583 6054 :
1583 6055 :
1583 6056 :
1583 6057 :
1583 6058 :
1583 6059 :
1583 6060 :
1583 6061 :
1583 6062 :
1583 6063 :
1583 6064 :
1583 6065 : IMPLICIT INPUTS:
1583 6066 :
1583 6067 : None.
1583 6068 :
1583 6069 : OUTPUT PARAMETERS:
1583 6070 :
1583 6071 : None.
```

2	
SIGNL ARY PNT	
MECH ARY PNT	
4	
ESTABLISH FP	
DEPTH	
R0	
R1	
N	
CONDITION NAME	
N-3 ADDITIONAL LONG WORD ARGS	
PC	
PSL	

Mechanism Array

Signal Array

```
15B3 6072 :  
15B3 6073 : IMPLICIT OUTPUTS:  
15B3 6074 :  
15B3 6075 :     If the call was due to a system error, then the CASE$V_ABORT bit in  
15B3 6076 :     CONTROL_CASE will be set to start the termination process.  
15B3 6077 :  
15B3 6078 : COMPLETION CODES:  
15B3 6079 :  
15B3 6080 :     $$$_CONTINUE - return to what we were doing  
15B3 6081 :  
15B3 6082 : SIDE EFFECTS:  
15B3 6083 :  
15B3 6084 :     If the call was due to a LIB$SIGNAL, then just write out the message.  
15B3 6085 :     If the call was due to an error, write out the error and start the  
15B3 6086 :     termination processes of the TSTCNTRL.  
15B3 6087 :  
15B3 6088 :     If an error occurs while in abort phase; then the any currently  
15B3 6089 :     running process is not stopped, and when it does stop it may be  
15B3 6090 :     left in the MWAIT state.  
15B3 6091 :  
15B3 6092 :--  
15B3 6093 :  
15B3 6094 : SSERROR:  
15B3 6095 : .WORD  ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; Entry mask  
15B5 6096 :  
6D  FB AF  DE 15B5 6097 : MOVAL  SSERROR,(FP) ; re-declare the exception handler  
15B9 6098 :  
15B9 6099 : $SETAST_S ENBFLG = #0 ; disable AST delivery  
09  5A  D4 15C2 6100 : CLRL   R10 ; assume it was disabled  
09  50  D1 15C4 6101 : CMPL   R0,#$$$_WASSET ; were AST's enabled?  
03  12 15C7 6102 : BNEQ   10$ ; no, so continue  
5A  01  D0 15C9 6103 : MOVL   #1,R10 ; save it to be reenabled  
15CC 6104 :  
15CC 6105 10$:  
15CC 6106 : $SETSFH_S ENBFLG = #0 ; Disable SS failure mode for PUTMSG  
09  5B  D4 15D5 6107 : CLRL   R11 ; assume it was disabled  
09  50  D1 15D7 6108 : CMPL   R0,#$$$_WASSET ; was SS failure mode enabled?  
03  12 15DA 6109 : BNEQ   20$ ; no, so continue  
5B  01  D0 15DC 6110 : MOVL   #1,R11 ; save it to be reenabled  
15DF 6111 :  
15DF 6112 20$:  
57  0C0B'CF DE 15DF 6113 : MOVAL  CNTRL_LOG_RAB,R7 ; get addr of perm log RAB  
15E4 6114 :  
56  04 AC  D0 15E4 6115 : MOVL   CHF$S_SIGARGST(AP),R6 ; Get the signal array pointer  
0C  10  ED 15E8 6116 : CMPZV  #ST$S$V_FAC_NO,#ST$S$S_FAC_NO,- ; Is this a message from LIB$SIGNAL?  
000000C3 8F 04 A6 15EB 6117 : CHF$S_SIG_NAME(R6),#TSTCNTRL_K  
1F  12 15F2 6118 : BNEQ   30$ ; BR if this is a system exception  
15F4 6119 :  
15F4 6120 : ; set facility code to be new one  
15F4 6121 :  
10  009E'CF F0 15F4 6122 : INSV   FAC_CODE,#ST$S$V_FAC_NO,- ; set the facility code of the  
0C  15F9 6123 : #ST$S$S_FAC_NO,- ; message to be the defined  
04  A6 15FA 6124 : CHF$S_SIG_NAME(R6) ; facility code  
15FC 6125 :  
66  02  C2 15FC 6126 : SUBL2  #2,CHF$S_SIG_ARGS(R6) ; Drop the PC and PSL  
15FF 6127 : $PUTMSG_S MSGVEC= CHF$S_SIG_ARGS(R6),- ; Print the message  
15FF 6128 : ACIRTN = LNG_FORMAT
```

```
00DB 31 1610 6129 BRW END_SSERROR ; Return to the program
      1610 6130
      1613 6131
      1613 6132 30$:
0000045C 8F D1 1613 6133 CMPL #SS$ SSFAIL,- ; see if system service failure
      04 A6 1617 6134 CHF$C SIG_NAME(R6) ; (which means it might be RMS)
      0B 12 161B 6135 BNEQ SYS_EXCEPT ; it is not an RMS error
      161D 6136
      0C 10 ED 161D 6137 CMPZV #ST$SV FAC_NO,#ST$SS FAC_NO,- ; Is it an RMS failure?
      01 08 A6 1620 6138 CHF$C SIG_ARG1(R6),#RMS_R
      03 12 1623 6139 BNEQ SYS_EXCEPT ; BR if not
      1625 6140
      00C6 31 1625 6141 BRW END_SSERROR ; Return to the program
      1628 6142
      1628 6143
      1628 6144 SYS_EXCEPT:
      1628 6145 ; we have a system error, see if we are in an abort state. If we are,
      1628 6146 ; then we cannot clean up anymore, so we exit. If we are not in an
      1628 6147 ; abort state, then we will print an error and start the abort process
      1628 6148 ; by calling the exit handler.
      1628 6149
      26 04A3'CF 0D E3 1628 6150 BBSC #TCNTRL$V_ABORT,FLAGS,10$ ; if not in abort, print error
      162E 6151
      162E 6152 ; we are in a abort sequence, cancel the exit handler and bail out
      162E 6153
      162E 6154 $SET$FM_S ENBFLG = R11 ; set to previous state
      1637 6155 $SET$ST_S ENBFLG = R10 ; set to previous state
      1640 6156
      1640 6157 $CANEXH_S ; cancel our exit handler . . .
      1649 6158 $EXIT_S CODE = STATUS ; . . . so that the system can
      1654 6159 ; worry about cleaning up
      1654 6160
      1654 6161 10$:
      1654 6162 ; now produce the reason for the error
      1654 6163
      04DA'CF 04 A6 D0 1654 6164 MOVL CHF$C SIG_NAME(R6),STATUS ; Save the status
      58 D4 165A 6165 CLRL R8 ; Assume that it is not SS failure
      04DA'CF 0000045C 8F D1 165C 6166 CMPL #SS$ SSFAIL,STATUS ; But, is it a SS failure?
      42 12 1665 6167 BNEQ 30$ ; No, so no special message
      1667 6168 $GETMSG_S MSGID = CHF$C SIG_ARG1(R6),- ; Get message for failure code
      1667 6169 MSGLEN = GETMSG_PTR,-
      1667 6170 BUFADR = GETMSG_DESC,-
      1667 6171 FLAGS = #14,- ; but without the text
      1667 6172 OUTADR = MSG_BLOCK
      167F 6173
      0287'CF 95 167F 6174 TSTB MSG_BLOCK+1 ; any FAO arguments?
      1E 13 1683 6175 BEQL 20$ ; no, so skip alternate message
      05C9'CF DF 1685 6176 PUSHAL GETMSG_PTR ; write message w/out FAO text
      01 DD 1689 6177 PUSHL #1
      00C31130 8F DD 168B 6178 PUSHL #TCNTRL$ TEXT ; put the message with an info messa
      08 A6 F0 1691 6179 INSV CHF$C SIG_ARG1(R6),- ; and put correct severity code
      00 1694 6180 #ST$SV SEVERITY,- ; with the message
      6E 03 1695 6181 #ST$SS SEVERITY,(SP)
      1697 6182
      10 009E'CF F0 1697 6183 INSV FAC_CODE,#ST$SV FAC_NO,- ; set the facility code of the
      0C 169C 6184 #ST$SS FAC_NO,- ; message to be the defined
      6E 169D 6185 (SP) ; facility code
```

```

      58 03 D0 169E 6186
      06 11 169E 6187      MOVL #3,R8      ; number of arguments we pushed
      16A1 6188      BRB 30$      ; leave
      16A3 6189
      16A3 6190 20$:
      08 A6 DD 16A3 6191      PUSHL CHF$_SIG_ARG1(R6)      ; no FA0 text, so write entire mesg.
      58 01 D0 16A6 6192      MOVL #1,R8      ; pushed one argument
      16A9 6193
      16A9 6194 30$:
      57 66 04 C5 16A9 6195      MULL3 #4,CHF$_SIG_ARGS(R6),R7      ; Convert longwords to bytes
      5E 57 C2 16AD 6196      SUBL2 R7,SP      ; Save the current signal array...
      6E 04 A6 57 28 16B0 6197      MOVCL3 R7,CHF$_SIG_NAME(R6),(SP)      ; ...on the stack
      16B5 6198
      59 66 58 C1 16B5 6199      ADDL3 R8,CHF$_SIG_ARGS(R6),R9      ; calculate the current number
      16B9 6200      ; of arguments
      16B9 6201
      59 04 C0 16B9 6202      ADDL2 #4,R9      ; add the following arguments:
      16BC 6203
      05: 00 DD 16BC 6204      PUSHL #0      ; the time parameter
      0F DF 16BE 6205      PUSHAL PRC_NAM_DESC      ; the process name
      02 DD 16C2 6206      PUSHL #2
      00C310E4 8F DD 16C4 6207      PUSHL #TCNTRL$ ABENDD!STSSK_SEVERE      ; a severe problem
      00000000'GF 59 FB 16CA 6208      CALLS R9,G^LIB$SIGNAL      ; write the message
      16D1 6209
      16D1 6210      $SETSFM_S ENBFLG = R11      ; set to previous state
      16DA 6211      $SETAST_S ENBFLG = R10      ; set to previous state
      16E3 6212
      16E3 6213      ; we want to start abort sequence, so we will do so by going through
      16E3 6214      ; the exit handler
      16E3 6215
      16E3 6216      $EXIT_S CODE = STATUS      ; start abort via exit handler
      16EE 6217
      16EE 6218      END_SSERROR:
      16EE 6219
      16EE 6220      $SETSFM_S ENBFLG = R11      ; set to previous state
      16F7 6221      $SETAST_S ENBFLG = R10      ; set to previous state
      1700 6222
      50 01 D0 1700 6223      MOVL #SS$_CONTINUE,R0      ; return to what we were doing
      1703 6224
      04 1703 6225      RET
```

```
000C 1704 6227 LNG_FORMAT:
      1704 6228 .WORD ^M<P2,R3>
      1706 6229
      1706 6230 ; prints the time stamps to the logfile
      1706 6231
      53 0C08'CF DE 1706 6232 MOVAL CNTRL_LOG_RAB,R3 ; get addr of perm log RAB
      52 04 AC D0 1708 6233 MOVL 4(AP),R2 ; get the message descriptor address
      22 A3 62 B0 170F 6234 MOVW (R2),RAB$W_RSZ(R3) ; put the size in
      28 A3 04 A2 D0 1713 6235 MOVL 4(R2),RAB$[RBF(R3) ; put the address in
      50 01 D0 1718 6236 $PUT RAB = R3 ; write the record
      0D 04A3'CF 05 E0 1721 6237 MOVL #SS$ NORMAL,R0 ; set return status
      07 04A3'CF 03 E0 1724 6238 BBS #TCNTRL$V_WRT_MSG,FLAGS,10$ ; print msg if suppose to
      50 D4 172A 6239 BBS #TCNTRL$V_LONG_REPORT,FLAGS.10$ ; print msg if report is long
      1730 6240
      1730 6241 CLRL R0 ; else, don't print msg
      04A3'CF 05 C8 1732 6242 BISL2 #TCNTRL$V_WRT_MSG,FLAGS ; turn wrt msg back on
      1737 6243
      1737 6244 10$:
      04 1737 6245 RET ; keep working
      1737 6246
```

```
1738 6248 .SBTTL RMS Error Handler
1738 6249 :++
1738 6250 : FUNCTIONAL DESCRIPTION:
1738 6251 :
1738 6252 :     This routine handles error returns from RMS calls.
1738 6253 :
1738 6254 : CALLING SEQUENCE:
1738 6255 :
1738 6256 :     Called by RMS when a file processing error is found.
1738 6257 :
1738 6258 : INPUT PARAMETERS:
1738 6259 :
1738 6260 :     None.
1738 6261 :
1738 6262 : IMPLICIT INPUTS:
1738 6263 :
1738 6264 :     The FAB or RAB associated with the RMS call.
1738 6265 :
1738 6266 : OUTPUT PARAMETERS:
1738 6267 :
1738 6268 :     None.
1738 6269 :
1738 6270 : IMPLICIT OUTPUTS:
1738 6271 :
1738 6272 :     If the error is severe, the CASE$V_ABORT bit will be set to start
1738 6273 :     TSTCNTRL termination.
1738 6274 :
1738 6275 : COMPLETION CODES:
1738 6276 :
1738 6277 :     SS$_NORMAL
1738 6278 :
1738 6279 : SIDE EFFECTS:
1738 6280 :
1738 6281 :     Program may start termination sequence depending on severity of the
1738 6282 :     error RMS errors RMS$_FNF and RMS$_EOF are ignored.
1738 6283 :
1738 6284 : --
1738 6285 :
1738 6286 RMSERR:
1738 6287 .WORD ^M<R2,R3,R4,R5,R6> ; Entry mask
1738 6288
1738 6289 MOVL 4(AP),R2 ; See whether we're dealing with...
1738 6290 CMPB #FAB$C_BID,FAB$B_BID(R2) ; ...a FAB or a RAB
1738 6291 BNEQ RAB_ERROR ; BR if it's a RAB
1738 6292
1738 6293 FAB_ERROR:
1738 6294 ; error was caused in by the FAB, see if it is acceptable, write
1738 6295 ; message if it is not
1738 6296
1738 6297 CMPL #RMS$_FNF,FAB$L_STS(R2) ; FNF is acceptable,
1738 6298 BNEQ 10$ ; it isn't FNF, continue
1738 6299 BRW END_RMSERR ; it is FNF, so leave
1738 6300
1738 6301 10$:
1738 6302 MOVAL FILE,R3 ; FAB-specific code: text string...
1738 6303 MOVL R2,R4 ; ...address of FAB...
1738 6304 PUSHL FAB$L_STV(R2) ; ...STV field for error...
```

007C

52 04 AC D0 173A 6288

62 03 91 173E 6290

21 12 1741 6291

1743 6292

1743 6293

1743 6294

1743 6295

1743 6296

08 A2 00000000'8F D1 1743 6297

03 12 174B 6298

00A4 31 174D 6299

1750 6300

1750 6301

53 031A'CF DE 1750 6302

54 52 D0 1755 6303

0C A2 DD 1758 6304

```

55 08 A2 DD 175B 6305      PUSHL  FAB$$_STS(R2)      ; ...STS field for error...
    08 A2 DO 175E 6306      MOVL   FAB$$_STS(R2),R5    ; ...and save the error code
    20 11 1762 6307      BRB    COMMON                ; FAB and RAB share other code
    1764 6308
    1764 6309 RAB_ERROR:
    1764 6310      ; error was caused in by the RAB, see if it is acceptable, write
    1764 6311      ; message if it is not
    1764 6312
08 A2 00000000'8F D1 1764 6313      CML  #RMS$_EOF,RAB$$_STS(R2) ; EOF is acceptable
    03 12 176C 6314      BNEQ  10$                     ; not EOF, so write error
    0083 31 176E 6315      BRW   END_RMSERR              ; it is EOF, so leave
    1771 6316
    1771 6317 10$:
    53 0326'CF DE 1771 6318      MOVAL  RECORD,R3          ; RAB-specific code: text string...
    54 3C A2 DO 1776 6319      MOVL   RAB$$_FAB(R2),R4      ; ...address of associated FAB...
    0C A2 DD 177A 6320      PUSHL  RAB$$_STV(R2)          ; ...STV field for error...
    08 A2 DD 177D 6321      PUSHL  RAB$$_STS(R2)          ; ...STS field for error...
    55 08 A2 DO 1780 6322      MOVL   RAB$$_STS(R2),R5      ; ...and save the error code
    1784 6323
    1784 6324 COMMON:
    1784 6325      ; we have an RMS error, see if we are in an abort state. If we are,
    1784 6326      ; then we cannot clean up anymore, so we exit. If we are not in an
    1784 6327      ; abort state, then we will print an error and start the abort process
    1784 6328      ; by calling the exit handler.
    1784 6329
14 04A3'CF OD E1 1784 6330      BBC    #TCNTRL$_ABORT,FLAGS,10$ ; if not in abort, print error
    178A 6331
    178A 6332      ; we are in a abort sequence, cancel the exit handler and bail out
    178A 6333
    178A 6334
    178A 6335
    1793 6336      $CANEXH_S                                ; cancel our exit handler . . .
    179E 6337      $EXIT_S_CODE = STATUS                    ; . . . so that the system can
    179E 6338 10$:
    56 34 A4 9A 179E 6339      MOVZBL  FAB$$_FNS(R4),R6
    17A2 6340      $FAO_S  CTRSTR = RMS_ERR_STRING,-      ; Common code, prepare error message
    17A2 6341      OUTLEN = BUFFER_PTR,-
    17A2 6342      OUTBUF = FAO_BUF,-
    17A2 6343      P1      = R3,-
    17A2 6344      P2      = R6,-
    17A2 6345      P3      = FAB$$_FNA(R4)
    0535'CF DF 17BC 6346      PUSHAL  BUFFER_PTR          ; ...and arguments for ERROR_EXIT...
    01 DD 17C0 6347      PUSHL  #1                      ; ...
00C31130 8F DD 17C2 6348      PUSHL  #TCNTRL$_TEXT        ; ...
    17C8 6349
10 009E'CF FO 17C8 6350      INSV   FAC_CODE,#STSSV_FAC_NO,- ; set the facility code of the
    0C 17CD 6351      #STSSS_FAC_NO,-                    ; message to be the defined
    6E 17CE 6352      (SP)                                ; facility code
    17CF 6353
    55 FO 17CF 6354      INSV   R5,-                      ; get the right serverity
    17D1 6355      #STSSV_SEVERITY,-
    03 00 17D1 6356      #STSSS_SEVERITY,-
    6E 17D3 6357      (SP)
    17D4 6358
    00 DD 17D4 6359      PUSHL  #0                        ; the time parameter
0516'CF DF 17D6 6360      PUSHAL  PRC_NAM_DESC           ; the process name
    02 DD 17DA 6361      PUSHL  #2
```

TSTCNTRL
V04-000

TEST PACKAGE CONTROL PROGRAM
RMS Error Handler

G 12

16-SEP-1984 01:30:05 VAX/VMS Macro V04-00
5-SEP-1984 04:26:03 [UETP.SRC]UETPHAS00.MAR;1

Page 147
(73)

```
00C310E4 8F DD 17DC 6362      PUSHL  #TCNTRL$ ABENDD!STSSK_SEVERE      ; a severe problem
00000000'GF 09 FB 17E2 6363      CALLS  #9,G^LIB$SIGNAL          ; write the message
                17E9 6364
                17E9 6365      ; we want to start abort sequence, so we will do so by going through
                17E9 6366      ; the exit handler
                17E9 6367
                17E9 6368      $EXIT_S CODE = STATUS                ; start abort via exit handler
                17F4 6369
                17F4 6370 END_RMSERR:
                17F4 6371
50 01 D0 17F4 6372      MOVL  #SS$ _NORMAL,R0                ; return success
                17F7 6373
04 17F7 6374      RET
```

```
17F8 6376      .SBTTL CTRL/C Handler
17F8 6377      .DEFAULT DISPLACEMENT, LONG
17F8 6378      :++
17F8 6379      : FUNCTIONAL DESCRIPTION:
17F8 6380      :
17F8 6381      : This routine is reached when a Control-C is typed by the user signaling
17F8 6382      : to abort the TSTCNTRL. The routine activates a repeating control-C
17F8 6383      : handler (via an out-of-band QIO) to "eat" up any further control-C's
17F8 6384      : which the user may type. A message is printed saying that the TSTCNTRL
17F8 6385      : has been aborted due to a control_C and the CASE$V_ABORT flag is set
17F8 6386      : to start the termination process.
17F8 6387      :
17F8 6388      : CALLING SEQUENCE:
17F8 6389      :
17F8 6390      : Called via AST
17F8 6391      :
17F8 6392      : INPUT PARAMETERS:
17F8 6393      :
17F8 6394      : None.
17F8 6395      :
17F8 6396      : IMPLICIT INPUTS:
17F8 6397      :
17F8 6398      : None.
17F8 6399      :
17F8 6400      : OUTPUT PARAMETERS:
17F8 6401      :
17F8 6402      : None.
17F8 6403      :
17F8 6404      : IMPLICIT OUTPUTS:
17F8 6405      :
17F8 6406      : The CASE$V_ABORT flag will be set to start the termination process.
17F8 6407      :
17F8 6408      : COMPLETION CODES:
17F8 6409      :
17F8 6410      : SSS_NORMAL
17F8 6411      :
17F8 6412      : SIDE EFFECTS:
17F8 6413      :
17F8 6414      : All further Control-C's will be ignored.
17F8 6415      : Termination of the TSTCNTRL will begin.
17F8 6416      :
17F8 6417      :--
17F8 6418      :
17F8 6419      : CCASTHAND:
OFFC 17F8 6420      .WORD  ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; Entry mask
17FA 6421
17FA 6422      ; set out-of-band so as to ignore any future Control-C's
17FA 6423
17FA 6424      $QIO_S CHAN = TTYCHAN, - ; do a read from the TTY
17FA 6425      FUNC = #IOS$ SETMODE!IOSM_OUTBAND, - ; look at the control character
17FA 6426      P1 = RPT CNTRLC AST, - ; and handle it if it is a ^C
17FA 6427      P2 = #CNTRLC_MSR
1823 6428
1823 6429      ; write abort message
1823 6430
0000016D'EF DF 1823 6431      PUSHAL CNTRLCMSG ; Set message pointer
01 DD 1829 6432      PUSHL #1 ; Set arg count
```

```
00C31130 8F DD 1828 6433 PUSHL #TCNTRL$_TEXT!STSSK_WARNING ; Set signal name
10 0000009E'EF F0 1831 6434
OC 1831 6435 INSV FAC_CODE,#STSSV_FAC_NO,- ; set the facility code of the
6E 1838 6436 #STSSS_FAC_NO,- ; message to be the defined
1839 6437 ; facility code
183A 6438
00 DD 183A 6439 PUSHL #0 ; Indicate an abnormal termination
00000516'EF DF 183C 6440 PUSHAL PRC_NAM_DESC
02 DD 1842 6441 PUSHL #2
00C310E0 8F DD 1844 6442 PUSHL #TCNTRL$_ABENDD!STSSK_WARNING ; Say abort with the time
00000000'GF 07 FB 184A 6443 CALLS #7,G^LIB$SIGNAL ; Output the message
1851 6444
1851 6445 ; set case to abort
1851 6446
0000049B'EF 01 C8 1851 6447 BISL2 #CASE$_ABORT,CONTROL_CASE ; start termination sequence
1858 6448
1858 6449 $WAKE_S ; wake in case we are in HIBER
1863 6450
50 01 D0 1863 6451 MOVL #SS$_NORMAL,R0 ; return success
04 1866 6452
1867 6453 RET
1867 6454
0000 1867 6455 RPT_CNTRLC_AST:
1869 6456 .WORD ^M<>
1869 6457 ; enter this AST if repeated CNTRL-C's are issued. To handle the
04 1869 6458 ; additional CNTRL-C, simply exit.
RET
```

```
186A 6461 .SBTTL Exit Handler
186A 6462 :++
186A 6463 : FUNCTIONAL DESCRIPTION:
186A 6464 :
186A 6465 : This routine is invoke only if the TSTCNTRL is stopped by external means
186A 6466 : (e.g., another process doing $FORCEX on it). The routine is called as
186A 6467 : an AST. When invoke, it will set the CASE$V ABORT bit to start
186A 6468 : termination of the TSTCNTRL, set the TCNTRL$V EXIT HAND bit to signal
186A 6469 : that we executed the exit handler, clear the AST that brought us here
186A 6470 : (this is so that we may receive termination AST's), and do a branch
186A 6471 : subroutine to the main case loop so as to receive the terminating
186A 6472 : processes. Right before the TSTCNTRL is about to exit, it will check
186A 6473 : to see if the TCNTRL$V_EXIT_HAND bit is set, and if it is, control will
186A 6474 : then return to this routine via a RSB.
186A 6475 :
186A 6476 : CALLING SEQUENCE:
186A 6477 :
186A 6478 : Reached as an AST when termination is attempted by outside mechanism.
186A 6479 :
186A 6480 : INPUT PARAMETERS:
186A 6481 :
186A 6482 : None.
186A 6483 :
186A 6484 : IMPLICIT INPUTS:
186A 6485 :
186A 6486 : None.
186A 6487 :
186A 6488 : OUTPUT PARAMETERS:
186A 6489 :
186A 6490 : None.
186A 6491 :
186A 6492 : IMPLICIT OUTPUTS:
186A 6493 :
186A 6494 : TCNTRL$V_EXIT_HAND bit is set in FLAGS.
186A 6495 : CASE$V_ABORT bit is set in CONTROL_CASE.
186A 6496 :
186A 6497 : COMPLETION CODES:
186A 6498 :
186A 6499 : TCNTRL$ _ABORT
186A 6500 :
186A 6501 : SIDE EFFECTS:
186A 6502 :
186A 6503 : Will start the termination process.
186A 6504 : Will clear the AST which brought us here.
186A 6505 : Will send control to the main case loop
186A 6506 :
186A 6507 :--
186A 6508 :
0000 186A 6509 EXIT_HANDLER:
186A 6510 .WORD ^M<> ; Entry mask
186C 6511
186C 6512 ; set the proper flags
186C 6513
186C 6514 BISL2 #CASE$M_ABORT,CONTROL_CASE ; to start termination process
000004A3'EF 00002000 8F C8 1873 6515 BISL2 #TCNTRL$M_ABORT,FLAGS ; say we are in abort mode
000004A3'EF 00008000 8F C8 187E 6516 BISL2 #TCNTRL$M_EXIT_HAND,FLAGS ; to say we came through the
1889 6517 ; exit handler
```

			1889	6518	
			1889	6519	
			1889	6520	
00000000'GF	00	FB	1889	6521	; clear the AST which brought us here
			1890	6522	
			1890	6523	
			1890	6524	; and send control back to the main case loop
	EAS5	30	1890	6525	
			1893	6526	BSBW MAIN_LOOP ; start termination
			1893	6527	
			1893	6528	; set our error status and leave for good
50	00C3002C	8F	1893	6529	
		D0	189A	6530	MOVL #TCNTRL\$_ABORT,R0 ; abort status
		04	189A	6531	
			189B	6532	RET ; BYE NOW!
			189B	6533	.END TSTCNTRL

TSTCNTRL
Symbol table

TEST PACKAGE CONTROL PROGRAM

L 12

16-SEP-1984 01:30:05 VAX/VMS Macro V04-00
5-SEP-1984 04:26:03 [UETP.SRC]UETPHAS00.MAR;1

Page 152
(75)

\$\$\$AST	= 00000001			AVL_LST_HEAD	= 00000018		
\$\$\$CNT	= 00000003			BAD_STATUS	0000022F	R	02
\$\$\$FLG	= FFFFFFFF			BASPRI	0000044E	R	04
\$\$\$KEY	= 0000000B			BIOLM	00000464	R	04
\$\$\$KFG	= FFFFFFFF			BUFFER	0000053D	R	04
\$\$\$MOD	= 00000000			BUFFER_PTR	00000535	R	04
\$\$\$TMP	= 0000003A	R	0D	BYTLM	00000469	R	04
\$.TAB	= 00000CE0	R	04	CASE\$K_CNTRL_CASE_SIZE	= 00000008		
\$.TABEND	= 00000D30	R	04	CASE\$M_ABORT	= 00000001		
\$.TMP	= 00000000			CASE\$M_ASSIGN	= 00000020		
\$.TMP1	= 00000001			CASE\$M_CREATE_PROC	= 00000040		
\$.TMP2	= 00000053			CASE\$M_EOF	= 00000008		
\$\$KEYTAB	= 00000000	R	0C	CASE\$M_FILE	= 00000010		
\$\$T1	= 00000001			CASE\$M_PARSE	= 00000080		
\$\$T2	= 00000006			CASE\$M_PROC_TERM	= 00000004		
..AFLG	= 00000000			CASE\$M_TIME_EXP	= 00000002		
..FLG	= 00000001			CASE\$S_CNTRL_CASE	= 00000004		
..TYP	= 000000CF			CASE\$V_ASSIGN	= 00000005		
.LEN	= 00000001			CASE\$V_CREATE_PROC	= 00000006		
ABORT_MSG	000000F0	R	02	CASE\$V_EOF	= 00000003		
ABORT_PROCESSES	000013DB	R	0A	CASE\$V_FILE	= 00000004		
ABORT_TCNTL_BLK	0000031F	R	0A	CASE_FILE_TYPE	00000AA4	R	0A
ABORT_TCNTL_SBRTN	00000467	R	0A	CCASTHAND	000017F8	R	0A
ABRT_DELT_SCNDS	= 0000000F			CHCK_ERROR	0000067B	R	0A
ABRT_TIM_ID	= 00000002			CHCK_FILESPC	000000A0	R	0B
ABT_C_CASE	= 00000008			CHCK_STATUS	00000AF8	R	0A
ABT_FLAGS	= 00000004			CHFSL_SIGARGLST	= 00000004		
ACCSL_BIOCNT	= 0000003C			CHFSL_SIG_ARG1	= 00000008		
ACCSL_CPUTIM	= 0000002C			CHFSL_SIG_ARGS	= 00000000		
ACCSL_DIOCNT	= 00000040			CHFSL_SIG_NAME	= 00000004		
ACCSL_FINALSTS	= 00000004			CHK_ABT_PROC	00000611	R	0A
ACCSL_PAGEFLTS	= 00000030			CHK_CNTR_C	00000058	R	0A
ACCSL_PGFLPEAK	= 00000034			CHK_END_SEG	000010FE	R	0A
ACCSL_PID	= 00000008			CHK_LOG_FIL	000006E4	R	0A
ACCSL_WSPEAK	= 00000038			CLEAN_OUT	0000083E	R	0A
ACCSQ_LOGIN	= 00000048			CLISGET_VALUE	*****	X	0A
ACCSQ_TERMTIME	= 00000010			CLISPRESENT	*****	X	0A
ACTIVE_PROC_LIMIT	00000008	R	04	CLIS NEGATED	*****	X	0A
ACT_PROC_LIM	= 00000014			CNTR[CMMSG	0000016D	R	02
ASG_CNT_CAS	= 00000004			CNTRLC_MSK	00000414	R	02
ASSIGN\$R_ASGN_CASE_SIZ	= 00000007			CNTRL_LOG_FAB	000008B8	R	04
ASSIGN\$M_COM	= 00000010			CNTRL_LOG_RAB	00000C08	R	04
ASSIGN\$M_LOG	= 00000001			COMMAND_ITM_LST	000000A2	R	04
ASSIGN\$M_NAME	= 00000002			COMMENT_DESC	000006E9	R	04
ASSIGN\$M_PAR	= 00000040			COMMENT_STR	000006F1	R	04
ASSIGN\$M_START	= 00000004			COMMON	00001784	R	0A
ASSIGN\$M_STOP	= 00000008			COMPLETE_Q_HEAD	00004658	R	03
ASSIGN\$M_TIME	= 00000020			COM_ASG	00000EFA	R	0A
ASSIGN\$S_ASGN_CASE	= 00000004			COM_ASG_DSC	= 00000004		
ASSIGN\$V_LOG	= 00000000			COM_ASG_RTN	00000F6F	R	0A
ASSIGN_BCK	00000397	R	0A	COM_EXIT_DESC	0000044E	R	02
ASSIGN_CASE	000004A7	R	04	COM_FAB	00000C4C	R	04
ASSIGN_SBRTN	00000EA0	R	0A	COM_FILE	00000AE8	R	0A
ASSIGN_VALUE_DESC	000008B4	R	04	COM_IMAGE	00000473	R	02
ASSIGN_VALUE_STR	000008BC	R	04	COM_LINE_DESC	00000015		
ASTLM	0000045F	R	04	COM_LINE_STR	0000001D		
AVAIL_LIST_HEAD	00004650	R	03	COM_P	00001311	R	0A

U
T

TSTCNTRL
Symbol table

TEST PACKAGE CONTROL PROGRAM

M 12

16-SEP-1984 01:30:05 VAX/VMS Macro V04-00
5-SEP-1984 04:26:03 [UETP.SRC]UETPHAS00.MAR;1Page 153
(75)

COM_PARM	= 00000AF0	R	0A	DEL_COM_FAB	= 00000CE0	R	04
COM_PARM_LENGTH	= 0000011D			DETERMINE_TYPE	= 00001295	R	0A
COM_RAB	= 0C000C9C	R	04	DEVSV_TRM	= 00000002		
COM_STAT_DESC	= 00000430	R	02	DEVICE_DESC	= 0000088C	R	04
COM_STR	= 00000508	R	02	DIB\$K_LENGTH	= 00000074		
CONCURRENT_PROC	= 00000945	R	0A	DIB\$W_UNIT	= 0000000C		
CONTIN_MAIN	= 000003D7	R	0A	DIOLM	= 00000473	R	04
CONTROL_CASE	= 00000498	R	04	DIRECTORY_DESC	= 00000894	R	04
CONT_COPY	= 00001319	R	0A	DSC\$A_POINTER	= 00000004		
CONVERT_TIME	= 000012D0	R	0A	DSC\$K_CLASS_D	= 00000002		
COPY_FICESPEC	= 00001242	R	0A	DSC\$K_CLASS_S	= 00000001		
COPY_PARAM	= 000012F6	R	0A	DSC\$K_DTYPE_T	= 0000000E		
COPY_RCRD	= 0000077E	R	0A	DSC\$W_LENGTH	= 00000000		
CPULM	= 0000046E	R	04	DVIS_DEVCLASS	= 00000004		
CP_ID_NUMBER	= 00000008			DVIS_DEVNAM	= 00000020		
CP_PROCESS_NAME	= 00000004			END_ABORT_PROC	= 00001427	R	0A
CP_TEST_FILENAME	= 0000000C			END_ASSIGN_SBRTN	= 00000F1E	R	0A
CR	= 0000000D			END_COM_ASG_RTN	= 00000F94	R	0A
CREATE_PROC_BLK	= 000003A3	R	0A	END_COPY_PARAM	= 0000132A	R	0A
CREATE_PROC_CASE	= 0000049F	R	04	END_CREATE_PROC	= 00000BC5	R	0A
CREATE_PROC_LENGTH	= 00000045			END_CRT_COM_PARM	= 00000DC3	R	0A
CREATE_PROC_SBRTN	= 00000962	R	0A	END_EOF_SBRTN	= 0000090C	R	0A
CREPRC_LIS	= 0000042A	R	04	END_EXPND_REGION	= 000013DA	R	0A
CRTPRC\$K_CREAT_CASE_SIZE	= 00000005			END_FILE_SBRTN	= 0000095E	R	0A
CRTPRC\$M_COM	= 00000004			END_FIND_PROC_SLOT	= 00001597	R	0A
CRTPRC\$M_COM_PARM	= 00000008			END_GET_PARSE	= 0000120C	R	0A
CRTPRC\$M_EXE	= 00000001			END_LOG_ASG_RTN	= 00000FD7	R	0A
CRTPRC\$M_EXE_PARM	= 00000002			END_MAIN	= 000003DA	R	0A
CRTPRC\$M_NULL	= 00000010			END_NUKE_PROC	= 00001484	R	0A
CRTPRC\$S_CRT_PROC_CASE	= 00000004			END_PROCESS_TERM_SBRTN	= 00000872	R	0A
CRTPRC\$V_COM	= 00000002			END_PROC_TERM_AST	= 0000154A	R	0A
CRTPRC\$V_EXE	= 00000000			END_PRS_FILE	= 000012CF	R	0A
CRTPRC\$V_NULL	= 00000004			END_RMSERR	= 000017F4	R	0A
CRT_COM_PARM	= 00000C6E	R	0A	END_SSERROR	= 000016EE	R	0A
CRT_COM_PRC	= 00000C47	R	0A	END_TCNTL	= 000003F4	R	0A
CRT_EXE_PARM	= 00000BE6	R	0A	END_TIM_ASG_RTN	= 0000101A	R	0A
CRT_EXE_PRC	= 00000BC9	R	0A	ENQCM	= 00000478	R	04
CRT_PRC_CASE	= 00000014			EOF_BLK	= 00000367	R	0A
CRT_PRC_FLG	= 0000000C			EOF_FLG	= 00000008		
CURRENT_LIST_HEAD	= 00004660	R	03	EOF_SBRTN	= 000008C3	R	0A
CURR_LST_HEAD	= 0000001C			ERROR	= 0000043E	R	04
C_CNTRL_CASE	= 00000010			ER_LOCATION_OF_STRUCT	= 0000000C		
C_PARSE_FLGS	= 00000024			ER_NUMBER_OF_SCOTS	= 00000004		
C_PROC_RUN	= 00000008			ER_SIZE_OF_SCOT	= 00000008		
C_TOT_PRC_LIM	= 00000020			EXE_FILE	= 00000ABB	R	0A
DATA_FILE_FAB	= 00000A90	R	04	EXE_P	= 00001309	R	0A
DATA_FILE_RAB	= 00000AE0	R	04	EXE_PARM	= 00000AC3	R	0A
DATA_RCRD_DESC	= 0000065D	R	04	EXE_STR	= 00000504	R	02
DATA_RCRD_STR	= 00000665	R	04	EXIT_DESC	= 000004CA	R	04
DATSTRUC_ERR	= 000001E9	R	02	EXIT_HANDLER	= 0000186A	R	0A
DAT_FIL_KEY	= 00000000	RG	0C	EXPLICIT_BLANKS	= 00001210	R	0A
DAT_FIL_RAB	= 0000000C			EXPND_REGION	= 0000136B	R	0A
DAT_FIL_STATE	= 00000000	RG	0B	E_CNTRL_CASE	= C0000004		
DAT_PARSE_BLK	= 00000000			FAB\$B_BID	= 00000000		
DOS_TERM	= 00000042			FAB\$B_FNS	= 00000034		
DELIMIT	= 000009CC	R	04	FAB\$C_BID	= 00000003		
DELLOG_DESC	= 00000067	R	04	FAB\$C_BLN	= 00000050		

TSTCNTRL
Symbol table

TEST PACKAGE CONTROL PROGRAM

N 12

16-SEP-1984 01:30:05 VAX/VMS Macro V04-00
5-SEP-1984 04:26:03 [UETP.SRC]UETPHAS00.MAR;1Page 154
(75)

FABSC_SEQ	= 00000000		
FABSC_VAR	= 00000002		
FABSL_ALQ	= 00000010		
FABSL_DEV	= 00000040		
FABSL_FNA	= 0000002C		
FABSL_FOP	= 00000004		
FABSL_STS	= 00000008		
FABSL_STV	= 0000000C		
FABSV_CHAN_MODE	= 00000002		
FABSV_CR	= 00000001		
FABSV_FILE_MODE	= 00000004		
FABSV_LNM_MODE	= 00000000		
FABSW_GBC	= 00000048		
FAB_ERROR	00001743	R	0A
FAC_CODE	0000009E	R	04
FAC_DESC	00000018	R	04
FAC_IN	0000007E	R	04
FAO_BUF	0000052D	R	04
FILE	0000031A	R	02
FILENAME_DESC	0000089C	R	04
FILENAME_LNGTH	= 00000009		
FILE_BLK	0000037B	R	0A
FILE_EXT_LNGTH	= 00000004		
FILE_SBRTN	00000910	R	0A
FILE_SPEC	0000011E	R	0B
FILE_SPEC_DESC	0000077D	R	04
FILE_SPEC_STR	00000785	R	04
FILLM	0000047D	R	04
FILNAM_DESC	0000002D	R	04
FILNAM_ERR	0000018F	R	02
FILNAM_IN	00000086	R	04
FIL_FLG	= 00000008		
FIND_FILESPEC	000000C2	R	0B
FIND_PROC_SLOT	00001559	R	0A
FINISH_LINE	000000D6	R	0B
FLAGS	000004A3	R	04
FS_LIST_HEADER	= 00000004		
FS_SLOT_PID	= 00000008		
F_CNTRL_CASE	= 00000004		
F_PARSE_FLGS	= 0000000C		
F_PROC_RUN	= 00000010		
GENERIC_ASG	00000180	R	0B
GETCHN_BUF	000003B4	R	04
GETCHN_DESC	000003AC	R	04
GETMSG_BUF	000005D1	R	04
GETMSG_DESC	000005C1	R	04
GETMSG_PTR	000005C9	R	04
GETUIC	000004DE	R	04
GET_AGAIN	00001139	R	0A
GET_ASG_VAL	000012E2	R	0A
GET_COMMENT	0000132B	R	0A
GET_DATA_RCRD	00000267	R	0A
GET_DAT_RCRD	00001092	R	0A
GET_PARAM	000000EA	R	0B
GET_PARSE_BLK	000003CF	R	0A
GET_PARSE_LENGTH	= 00000254		
GET_PARSE_SBRTN	0000107D	R	0A

GET_QUAL	00000102	R	0A
GET_RCRD	00000729	R	0A
GET_REST	00000136	R	0B
GET_TIME_STRING	000001BC	R	0B
HIBER_BLK	00000315	R	0A
IDS_BEGIN_OF_STRUCT	= 0000000C		
IDS_LIST_HEADER	= 00000010		
IDS_NUMBER_OF_SLOTS	= 00000004		
IDS_SIZE_OF_SLOT	= 000C0008		
ID_ADDR	00000004		
ID_DESC	00000000		
ID_DESC_LENGTH	= 00000010		
ID_NUMB	0000000C		
ID_OVERHEAD_SIZ	= 00000002		
ID_STR	00000008		
ID_STR_SIZ	= 00000004		
IMAGE	00000432	R	04
IMPLICIT_BLANKS	00001217	R	0A
INC_COUNT	00000B5A	R	0A
INC_TOTAL_RUN	00000B90	R	0A
INDENT_LOG_RECORD	00000876	R	0A
INIT_STRUCT	0000134C	R	0A
INPUT	00000436	R	04
IN_DEVICE_CLASS	00000106	R	04
IN_DEVICE_DESC	000000BE	R	04
IN_DEVICE_STR	0000C0C6	R	04
IOSM_CTRLCAST	*****	X	0A
IOSM_NOW	*****	X	0A
IOSM_OUTBAND	*****	X	0A
IOS_READVBLK	*****	X	0A
IOS_SETMODE	*****	X	0A
IOS_WRITEVBLK	*****	X	0A
JPI\$ASTLM	= 00000409		
JPI\$BIOLM	= 00000310		
JPI\$BYTLM	= 0000031A		
JPI\$CPULIM	= 0000040D		
JPI\$DIOLM	= 00000313		
JPI\$ENQLM	= 00000320		
JPI\$FILLM	= 0000040F		
JPI\$PGFLQUOTA	= 0000040E		
JPI\$PRCLM	= 00000408		
JPI\$PRCNAM	= 0000031C		
JPI\$PRIB	= 00000309		
JPI\$TQLM	= 00000410		
JPI\$UIC	= 00000304		
JPI\$WSQUOTA	= 00000402		
LARGE_BUFFER	= 00000100		
LF	= 0000000A		
LGN_ASG_DSC	= 00000004		
LIB\$CVT-DTIME	*****	X	0A
LIB\$SIGNAL	*****	X	0A
LIB\$TPARSE	*****	X	0A
LIB\$SYNTAXERR	= 00158284		
LNAM_DESC	= 000000C4		
LNG_FORMAT	00001704	R	0A
LNMS_STRING	= 00000002		
LNMRP	= 0000000A		

TSTCNTRL
Symbol table

TEST PACKAGE CONTROL PROGRAM

B 13

16-SEP-1984 01:30:05 VAX/VMS Macro V04-00
5-SEP-1984 04:26:03 [UETP.SRC]UETPHAS00.MAR;1Page 155
(75)

LNMGRPLEN	= 00000010		NO_SET_LOG	000003D9	R	02	
LNMGRPNIL	0000050E	R	04	NO_SET_TIM	000002E0	R	02
LNMGRPNUM	000004FE	R	04	NUKE_PROC	00001453	R	0A
LNMITMLST	0000004C	R	02	NUKE_PROC_AST	0000143E	R	0A
LNMPRCDIR	00000012	R	02	NULL_FILE	00000AB8	R	0A
LNMTMPMBX	0000002F	R	02	NUMB_OF_SLOTS	= 00000032		
LOGFIL_ASG	00000156	R	0B	OTSSCVT_L_TI	*****	X	0A
LOG_ASG	00000ECA	R	0A	OTSSCVT_L_TO	*****	X	0A
LOG_ASG_RTN	00000F98	R	0A	OTSSCVT_TI_L	*****	X	0A
LOG_EXT_DESC	0000041C	R	02	OUTPUT	0000043A	R	04
LOG_FIL_DESC	00000173	R	04	PAGE_SIZE	= 00000200		
LOG_FIL_STR	0000017B	R	04	PARAM_DESC	00000940	R	04
LOG_RCRD_DESC	00000015			PARAM_STR	00000948	R	04
LOG_RCRD_STR	0000001D			PARAM_STRING	0000010E	R	0B
MAILBOX_SIZE	= 00000054			PARCNT_DESC	0000003E	R	04
MAIN_LOOP	000002E8	R	0A	PARCNT_IN	00000096	R	04
MAKE_PROC_NAM	00000DC7	R	0A	PARSES\$M_GET_NXT_RCRD	= 00000002		
MAKE_TMP_LOG_NAM	00000E35	R	0A	PARSES\$M_PROC_RUNS_OTHERS	= 00000001		
MAXTIME_ID	= 00000001			PARSES\$V_GET_NXT_RCRD	= 00000001		
MAX_TMP_LOG_NAM	= 00000005			PARSES\$V_PROC_RUNS_OTHERS	= 00000000		
MBXONT	00000456	R	04	PARSE_DAT_RCRD	000010DE	R	0A
MBX_CHAN	0000011D	R	04	PARSE_FLAGS	00000659	R	04
MBX_LOGNAM_DESC	0000002C			PAR_ASG	00000F12	R	0A
MBX_LOGNAM_STR	00000034			PAR_ASG_DSC	= 00000004		
MBX_UNIT	00000428	R	04	PAR_ASG_RTN	0000101E	R	0A
MF_ID_NUMBER	= 0000000C			PEN_FILE	000000CF	R	0A
MF_TEMP_EXT	= 00000008			PGF[QUOTA	00000482	R	04
MF_TEMP_FILE	= 00000004			PIDADR	0000042E	R	04
MF_TEST_FILENAME	= 00000010			PNAME_ASG	00000ED6	R	0A
MSG_BLOCK	00000286	R	04	PNAME_ASG_RTN	00000F22	R	0A
NAM\$B_DEV	= 00000039			PNAME_FILNAM_SIZ	= 00000009		
NAM\$B_DIR	= 0000003A			PNAME_ASG_DSC	= 00000004		
NAM\$B_ESL	= 0000000B			PQL\$ASTCM	= 00000001		
NAM\$B_ESS	= 0000000A			PQL\$BIOLM	= 00000002		
NAM\$B_NAME	= 0000003B			PQL\$BYTLM	= 00000003		
NAM\$B_NODE	= 00000038			PQL\$CPULM	= 00000004		
NAM\$B_NOP	= 00000008			PQL\$DIOLM	= 00000005		
NAM\$B_RSS	= 00000002			PQL\$ENQLM	= 0000000C		
NAM\$B_TYPE	= 0000003C			PQL\$FILLM	= 00000006		
NAM\$B_VER	= 0000003D			PQL\$LISTEND	= 00000000		
NAM\$C_BID	= 00000002			PQL\$PGFLQUOTA	= 00000007		
NAM\$C_BLN	= 00000060			PQL\$PRCLM	= 00000008		
NAM\$C_MAXRSS	= 000000FF			PQL\$TQELM	= 00000009		
NAM\$L_DEV	= 00000044			PQL\$WSDEFAULT	= 0000000B		
NAM\$L_DIR	= 00000048			PQL\$WSQUOTA	= 0000000A		
NAM\$L_ESA	= 0000000C			PRCS\$M_LOGIN	= 00000040		
NAM\$L_NAME	= 0000004C			PRCINFO\$A_BLINK	= 00000004		
NAM\$L_NODE	= 00000040			PRCINFO\$A_DEVICE	= 00000044		
NAM\$L_RSA	= 00000004			PRCINFO\$A_EXTENSION	= 00000050		
NAM\$L_TYPE	= 00000050			PRCINFO\$A_FILNAM	= 0000004C		
NAM\$L_VER	= 00000054			PRCINFO\$A_FLINK	= 00000000		
NAME_SIZE	= 00000009			PRCINFO\$A_NODE	= 00000040		
NBR_PDR_BLNKS	= 00000004			PRCINFO\$A_VERSION	= 00000054		
NIL_EXT_DESC	00000428	R	02	PRCINFO\$L_BIOCNT	= 00000028		
NODE_DESC	00000884	R	04	PRCINFO\$L_CPUTIM	= 00000018		
NON_DELIMITER	00000146	R	0B	PRCINFO\$L_DIOCNT	= 0000002C		
NO_CRT_PRC	00000264	R	02	PRCINFO\$L_FINALSTS	= 00000008		

TSTCNTRL
Symbol table

TEST PACKAGE CONTROL PROGRAM

C 13

16-SEP-1984 01:30:05 VAX/VMS Macro V04-00
5-SEP-1984 04:26:03 [UETP.SRC]UETPHAS00.MAR;1Page 156
(75)U
V

PRCINFO\$L_FLAGS = 00000038
PRCINFO\$L_PAGEFLTS = 0000001C
PRCINFO\$L_PGFLPEAK = 00000020
PRCINFO\$L_PID = 0000000C
PRCINFO\$L_WSPEAK = 00000024
PRCINFO\$M_MBX_CHAN = 00000001
PRCINFO\$M_PROC_ABORTED = 00000002
PRCINFO\$Q_LOGIN = 00000030
PRCINFO\$Q_TERMTIME = 00000010
PRCINFO\$S_PRC_INFO = 00000168
PRCINFO\$T_FILE_SPEC = 0000006C
PRCINFO\$V_MBX_CHAN = 00000000
PRCINFO\$V_PROC_ABORTED = 00000001
PRCINFO\$W_DEVICE_SIZ = 0000005C
PRCINFO\$W_DIR_SIZ = 0000005E
PRCINFO\$W_EXTENSION_SIZ = 00000062
PRCINFO\$W_FILESPEC_SIZ = 00000058
PRCINFO\$W_FILNAM_SIZ = 00000060
PRCINFO\$W_ID_NUMB = 0000003E
PRCINFO\$W_MBX_CHAN = 0000003C
PRCINFO\$W_NODE_SIZ = 0000005A
PRCINFO\$W_VERSION_SIZ = 00000064
PRCLM = 00000487 R 04
PRCNAM = 0000044A R 04
PRCNAMDESC = 00000483 R 04
PRCNAMSTR = 0000048B R 04
PRC_NAM_DESC = 00000516 R 04
PRC_NAM_STR = 0000051E R 04
PRIV MSR = 00000010 R 04
PROCESS_CMPLT = 000006FF R 0A
PROCESS_NAME_SIZ = 0000000F
PROCESS_RUNNING = 00000000 R 04
PROCESS_TERM_BLK = 00000343 R 0A
PROCESS_TERM_SBRTN = 00000506 R 0A
PROC_ACT = 00000394 R 02
PROC_INFO_TOP = 00000000 R 03
PROC_TERM_AST = 00001491 R 0A
PROC_TERM_DELT = 0000000A R 02
PROC_TERM_IOSB = 000004AB R 04
PROC_TERM_LENGTH = 00000259
PRSE_FILE\$PC_FAB = 000009E0 R 04
PRSE_FILE\$PC_NAM = 00000A30 R 04
PRS_FILE_SPEC = 0000121E R 0A
PRS_FILE_SUC = 000012CC R 0A
PRV\$V_DETACH = 00000005
PRV\$V_WORLD = 00000010
PRVADR = 00000442 R 04
P_AVL_LST_HEAD = 0000001C
P_CMPLT_Q_HEAD = 00000004
P_CNTRL_CASE = 00000010
P_PROCESS_RUN = 00000008
P_TERM_FLG = 0000000C
QUOTA = 00000446 R 04
QUOTA_LIS = 0000005C R 02
QUOTA_TABLE = 0000045E R 04
QUOTED_TIME_STRING = 000001CC R 0B
Q_STRING = 0000012E R 0B

RAB\$B_P\$Z = 00000034
RAB\$B_RAC = 0000001E
RAB\$C_BID = 00000001
RAB\$C_BLN = 00000044
RAB\$C_SEQ = 00000000
RAB\$C_CTX = 00000018
RAB\$C_FAB = 0000003C
RAB\$C_PBF = 00000030
RAB\$C_RBF = 00000028
RAB\$C_RBP = 00000004
RAB\$C_STS = 00000008
RAB\$C_STV = 0000000C
RAB\$C_UBF = 00000024
RAB\$V_PMT = 0000001E
RAB\$W_RSZ = 00000022
RAB_ERROR = 00001764 R 0A
RCRD_ERR_DESC = 00000024
RCRD_ERR_FAO = 0000002C
RCRD_ERR_MSG = 00000492 R 02
RCRD_ERR_STR = 00000034
RCRD_SIZE = 00000084
RECORD = 00000326 R 02
REPORT_DESC = 00000054 R 04
REPORT_ERR = 000001C4 R 02
REPORT_IN = 0000008E R 04
RETURN_BLK = 00000858 R 0A
RM\$S_EOF = ***** X 0A
RM\$S_FNF = ***** X 0A
RM\$S_TYP = ***** X 0A
RM\$ERR = 00001738 R 0A
RMS_ERR_STRING = 00000334 R 02
RMS_K = 00000001
RPT_CNTRL_C_AST = 00001867 R 0A
RTN_NAM_DESC = 00000242
RTN_NAM_STR = 0000024A
RUN_PROC = 00000090 R 0B
SEQUENTIAL_PROC = 00000925 R 0A
SHR\$ABEND0 = 000010E0
SHR\$BEGIN0 = 00001038
SHR\$ENDEDD = 00001080
SHR\$OPENIN = 00001098
SHR\$TEXT = 00001130
SLOT_NOT_FND = 0000157D R 0A
SLOT_TOP = 00000008
SPEC_DESC = 00000775 R 04
SS\$ABORT = 0000002C
SS\$CANCEL = 00000830
SS\$CONTINUE = 00000001
SS\$CONTROL_C = 00000651
SS\$FLTDIV = 00000494
SS\$NORMAL = 00000001
SS\$NOSUCHSEC = 00000978
SS\$SFAIL = 0000045C
SS\$WASSET = 00000009
SSERROR = 000015B3 R 0A
START = 00000000 R 0B
START_ASG = 00000EE2 R 0A

TSTCNTRL
Symbol table

TEST PACKAGE CONTROL PROGRAM

D 13

16-SEP-1984 01:30:05 VAX/VMS Macro V04-00
5-SEP-1984 04:26:03 [UETP.SRC]UETPHAS00.MAR;1

Page 157
(75)

START_ASG RTN	00000F43	R	0A	SYSSQIO	*****	GX	0A
START_DESC	0000028A	R	04	SYSSQIOW	*****	GX	0A
START_STR	00000292	R	04	SYSSREWIND	*****	GX	0A
START_STR_LEN	= 00000005			SYSSSETAST	*****	GX	0A
STATUS	000004DA	R	04	S'SSSETIMR	*****	GX	0A
STOP_ASG	00000EEE	R	0A	S'ISSSETPRN	*****	GX	0A
STOP_ASG RTN	00000F59	R	0A	SYSSSETPRV	*****	GX	0A
STOP_DESC	0000031B	R	04	SYSSSETSPM	*****	GX	0A
STOP_PROC	000013F0	R	0A	SYSSWAKE	*****	GX	0A
STOP_STR	00000323	R	04	SYS_COMMAND	0000010A	R	04
STOP_STR_LEN	= 00000005			SYS_EXCEPT	00001628	R	0A
STP_ASG_DSC	= 00000004			TCNTRL	= 00C30000		
STP_DESC	= 00000018			TCNTRL\$M_ABLE_TO_WRAP	= 00000010		
STR\$UPCASE	*****	X	0A	TCNTRL\$M_ABORT	= 00002000		
STRCT TOP BOT	0000003D			TCNTRL\$M_DELETE_TEMP_LOG	= 00000004		
STRT_ASG_DSC	= 00000004			TCNTRL\$M_EXIT	= 00C00800		
STRT_DESC	= 00000014			TCNTRL\$M_EXIT_HAND	= 00008000		
STRT_MSG	00000293	R	0A	TCNTRL\$M_FRST_FILE_REACHED	= 00000001		
STSSK_INFO	= 00000003			TCNTRL\$M_LONG_REPORT	= 00000008		
STSSK_SEVERE	= 00000004			TCNTRL\$M_PRNT_COMMENTS	= 00000040		
STSSK_SUCCESS	= 00000001			TCNTRL\$M_PROC_NUKED	= 00010000		
STSSK_WARNING	= 00000000			TCNTRL\$M_PROC_PEND	= 00001000		
STSSM_INHIB MSG	= 10000000			TCNTRL\$M_REWOOND	= 00000400		
STSSS_FAC NO	= 0000000C			TCNTRL\$M_SET_LOGNAM	= 00000080		
STSSS_SEVERITY	= 00000003			TCNTRL\$M_STRT MORE	= 00000002		
STSSV_FAC NO	= 00000010			TCNTRL\$M_TIMER SET	= 00000100		
STSSV_SEVERITY	= 00000000			TCNTRL\$M_TIME EXP	= 00004000		
STSFLG	0000045A	R	04	TCNTRL\$M_WRT MSG	= 00000020		
SYNTAX ERROR	00001145	R	0A	TCNTRL\$V_ABLE_TO_WRAP	= 00000004		
SYSS\$ASSIGN	*****	GX	0A	TCNTRL\$V_ABORT	= 0000000D		
SYSS\$CANEXH	*****	GX	0A	TCNTRL\$V_DELETE_TEMP_LOG	= 00000002		
SYSS\$CANTIM	*****	GX	0A	TCNTRL\$V_EXIT	= 0000000B		
SYSS\$CLOSE	*****	GX	0A	TCNTRL\$V_EXIT_HAND	= 0000000F		
SYSS\$CLRAST	*****	X	0A	TCNTRL\$V_FRST_FILE_REACHED	= 00000000		
SYSS\$CONNECT	*****	GX	0A	TCNTRL\$V_LONG_REPORT	= 00000003		
SYSS\$CREATE	*****	GX	0A	TCNTRL\$V_PRNT_COMMENTS	= 00000006		
SYSS\$CRELMN	*****	GX	0A	TCNTRL\$V_PROC_NUKED	= 00000010		
SYSS\$CREMBX	*****	GX	0A	TCNTRL\$V_PROC_PEND	= 0000000C		
SYSS\$CREPRC	*****	GX	0A	TCNTRL\$V_REWOOND	= 0000000A		
SYSS\$DASSGN	*****	GX	0A	TCNTRL\$V_SET_LOGNAM	= 00000007		
SYSS\$DCLEXH	*****	GX	0A	TCNTRL\$V_STRT MORE	= 00000001		
SYSS\$DELPRC	*****	GX	0A	TCNTRL\$V_TIMER SET	= 00000008		
SYSS\$ERASE	*****	GX	0A	TCNTRL\$V_TIME EXP	= 0000000E		
SYSS\$EXIT	*****	GX	0A	TCNTRL\$V_WRT MSG	= 00000005		
SYSS\$EXPREG	*****	GX	0A	TCNTRL\$X\$BEND	= 00C31CE0		
SYSS\$FAO	*****	X	0A	TCNTRL\$X_ABORT	= 00C3002C		
SYSS\$FORCEX	*****	GX	0A	TCNTRL\$X_BEGIN	= 00C31038		
SYSS\$GET	*****	GX	0A	TCNTRL\$X_CONTROLC	= 00C30651		
SYSS\$GETCHN	*****	GX	0A	TCNTRL\$X_ENDED	= 00C31080		
SYSS\$GETDVIW	*****	GX	0A	TCNTRL\$X_OPENIN	= 00C31098		
SYSS\$GETJPIW	*****	GX	0A	TCNTRL\$X_TEXT	= 00C31130		
SYSS\$GETMSG	*****	GX	0A	TCNTRL_X\$ABORT_MSG	0000013C	R	02
SYSS\$HIBER	*****	GX	0A	TCNTRL_K	= 000000C3		
SYSS\$OPEN	*****	GX	0A	TCNTRL_PRMT	00000000	R	02
SYSS\$PARSE	*****	GX	0A	TCNTRL_PRMT_SIZ	= 0000000A		
SYSS\$PUT	*****	GX	0A	TEMP_BUFF_DESC	0000011D		
SYSS\$PUTMSG	*****	GX	0A	TEMP_BUFF_STR	00000125		

U
V

5

5

4

2

4

2

2

6

7

TSTCNTRL
Symbol table

TEST PACKAGE CONTROL PROGRAM

E 13

16-SEP-1984 01:30:05 VAX/VMS Macro V04-00
5-SEP-1984 04:26:03 [UETP.SRC]UETPHAS00.MAR;1

Page 158
(75)

TEMP_COM_DESC	00000000		
TEMP_COM_STR	00000008		
TEMP_LOG_DESC	00000017		
TEMP_LOG_STR	0000001F		
TEMP_PNAM_DESC	00000000		
TEMP_PNAM_STR	00000008		
TERM_BUFF	0000011F	R	04
TIMER_EXP_BLK	0000032F	R	0A
TIME_ASG	00000192	R	0B
TIME_DESC	000009CD	R	04
TIME_EXP_AST	00001598	R	0A
TIME_EXP_SBR TN	000004A3	R	0A
TIME_OUT_ERR	000002A4	R	02
TIME_VALUE	000009D5	R	04
TIM_ASG	00000F06	R	0A
TIM_ASG RTN	00C00FDB	R	0A
TIM_C CASE	= 00000008		
TIM_FLAGS	= 00000004		
TIM_PRSE	= 0000000C		
TIM_VAL	= 00000004		
TMP_COM_EXT_DESC	00000467	R	02
TMP_LOG_FAB	00000B24	R	04
TMP_LOG_RAB	00000B74	R	04
TOKEN_ERR_DESC	0000013C		
TOKEN_ERR_FAO	00000144		
TOKEN_ERR_MSG	000004CD	R	02
TOKEN_ERR_STR	0000014C		
TOTAL_PROC_LIMIT	0000000C	R	04
TOTAL_PROC_RUN	00000004	R	04
TOT_PROC	= 00000004		
TOT_PROC RUN	= 00000004		
TPASB_CHAR	= 00000018		
TPASK_COUNT0	= 00000008		
TPASK_LENGTH0	= 00000024		
TPASL_COUNT	= 00000000		
TPASL_OPTIONS	= 00000004		
TPASL_STRINGCNT	= 00000008		
TPASL_STRINGPTR	= 0000000C		
TPASL_TOKENCNT	= 00000010		
TPASL_TOKENPTR	= 00C00014		
TPASM_ABBREV	= 00000002		
TPASM_BLANKS	= 00000001		
TPAS_ALPHA	= 000001EE		
TPAS_ANY	= 000001ED		
TPAS_BLANK	= 000001F2		
TPAS_DECIMAL	= 000001F3		
TPAS_DIGIT	= 000001EF		
TPAS_EOS	= 000001F7		
TPAS_EXIT	= FFFFFFFF		
TPAS_FAIL	= FFFFFFFE		
TPAS_FILESPEC	= 000001EA		
TPAS_HEX	= 000001F5		
TPAS_IDENT	= 000001EC		
TPAS_KEYWORD	= 00000100		
TPAS_LAMBDA	= 000001F6		
TPAS_MAXKEY	= 000000DC		
TPAS_OCTAL	= 000001F4		

TPAS_STRING	= 000001F0		
TPAS_SUBXPR	= 000001F8		
TPAS_SYMBOL	= 000001F1		
TPAS_UIC	= 000001EB		
TPARSE_ERR	000011A4	R	0A
TQELM	0000048C	R	04
TRMNTN_COM_DESC	00000000		
TRMNTN_COM_STR	00000008		
TRMNTN_END	0000081B	R	0A
TRMNTN_FIL_DESC	00000225		
TRMNTN_LOG_DESC	0000022D		
TRMNTN_LOG_STR	00000235		
TSTCNTRL	00000000	RG	0A
TST_DELIM	0000133F	R	0A
TTCHAN	00000655	R	04
TYPE_DESC	000008A4	R	04
UETPS_COPY_LOG_LINE	= 007480B9		
UIC	00000452	R	04
UIC_CODE	000004FA	R	04
VERSION_DESC	000008AC	R	04
WSDEFAULT	00000491	R	04
WSQUOTA	00000496	R	04
ZERO_PARCNT	00000355	R	02

U
V
6
6
2
6
4
6
2
2
7
2
4
6
6
6
4
6
6
6
2
6
7
6
2
6
7
6
2
2

+-----+
! Psect synopsis !
+-----+

PSECT name	Allocation	PSECT No.	Attributes
ABS	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
\$ABSS	00000000 (0.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
RODATA	0000050C (1292.)	02 (2.)	NOPIC USR CON REL LCL NOSHR NOEXE RD NOWRT NOVEC PAGE
PROCINFO	00004668 (18024.)	03 (3.)	NOPIC USR CON REL LCL NOSHR NOEXE RD WRT NOVEC PAGE
RWDATA	00000D30 (3376.)	04 (4.)	NOPIC USR CON REL LCL NOSHR NOEXE RD WRT NOVEC PAGE
ID_DESC_LOCAL	00000010 (16.)	05 (5.)	NOPIC USR CON ABS LCL NOSHR NOEXE RD NOWRT NOVEC PAGE
PROC_TERM_LOCAL	00000259 (601.)	06 (6.)	NOPIC USR CON ABS LCL NOSHR NOEXE RD NOWRT NOVEC PAGE
TEMP_COM_LOCAL	0000011D (285.)	07 (7.)	NOPIC USR CON ABS LCL NOSHR NOEXE RD NOWRT NOVEC PAGE
CREAT_PROC_LOCAL	00000045 (69.)	08 (8.)	NOPIC USR CON ABS LCL NOSHR NOEXE RD NOWRT NOVEC PAGE
GET_PARSE_LOCAL	00000254 (596.)	09 (9.)	NOPIC USR CON ABS LCL NOSHR NOEXE RD NOWRT NOVEC PAGE
TSTCNTRL	0000189B (6299.)	0A (10.)	NOPIC USR CON REL LCL NOSHR EXE RD NOWRT NOVEC PAGE
_LIB\$STATES	000001D4 (468.)	0B (11.)	PIC USR CON REL LCL SHR EXE RD NOWRT NOVEC BYTE
_LIB\$KEY0\$	00000018 (24.)	0C (12.)	PIC USR CON REL LCL SHR EXE RD NOWRT NOVEC WORD
_LIB\$KEY1\$	0000003D (61.)	0D (13.)	PIC USR CON REL LCL SHR EXE RD NOWRT NOVEC WORD

+-----+
! Performance indicators !
+-----+

Phase	Page faults	CPU Time	Elapsed Time
Initialization	36	00:00:00.08	00:00:00.82
Command processing	122	00:00:00.71	00:00:02.04
Pass 1	1804	00:01:02.17	00:02:19.99
Symbol table sort	10	00:00:04.09	00:00:09.48
Pass 2	1084	00:00:18.97	00:00:48.45
Symbol table output	108	00:00:00.65	00:00:01.39
Psect synopsis output	8	00:00:00.06	00:00:00.20
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	3175	00:01:26.76	00:03:22.41

The working set limit was 900 pages.

333455 bytes (652 pages) of virtual memory were used to buffer the intermediate code.

There were 150 pages of symbol table space allocated to hold 2707 non-local and 81 local symbols.

6533 source lines were read in Pass 1, producing 69 object records in Pass 2.

117 pages of virtual memory were used to define 96 macros.

+-----+
! Macro library statistics !
+-----+

Macro library name	Macros defined
\$255\$DUA28:[UETP.OBJ]UETP.MLB;1	2
\$255\$DUA28:[SYS.OBJ]LIB.MLB;1	0
\$255\$DUA28:[SYSLIB]STARLET.MLB;2	82
TOTALS (all libraries)	84

3363 GETS were required to define 84 macros.

TSTCNTRL
VAX-11 Macro Run Statistics

TEST PACKAGE CONTROL PROGRAM

G 13

16-SEP-1984 01:30:05 VAX/VMS Macro V04-00
5-SEP-1984 04:26:03 [UETP.SPC]UETPHAS00.MAR;1

Page 160
(75)

There were no errors, warnings or information messages.

MACRO/LIS=LISS:UETPHAS00/OBJ=OBJ\$:UETPHAS00 MSRC\$:UETPHAS00/UPDATE=(ENH\$:UETPHAS00)+EXECML\$/LIB+LIB\$:UETP/LIB

U
V

6
3

5
5

5

0412

AH-BT13A-SE
VAX/VMS V4.

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY